CORNEL I. NISTOR

CĂLIN GALERIU

PRELUCRAREA AUTOMATĂ A DATELOR FIZICE ȘI METODE NUMERICE

APLICAȚII ÎN TURBO PASCAL 6

EDITURA UNIVERSITĂȚII DIN BUCUREȘTI

- 1996 -

Conf. dr. CORNEL I. NISTOR CĂLIN GALERIU

PRELUCRAREA AUTOMATĂ A DATELOR FIZICE ŞI METODE NUMERICE

APLICAŢII ÎN TURBO PASCAL 6

EDITURA UNIVERSITĂŢII BUCUREŞTI
-1996-

Referenţi ştiinţifici: Prof. dr. VOICU GRECU
Conf. dr. ANIŞOARA CONSTANTINESCU

Toate drepturile de autor sunt rezervate Editurii Universităţii Bucureşti.
Orice reproducere sau traducere, fie şi parţială, precum şi contrafacerile de orice tip, intră sub incidenţa legii penale.

ISBN-973-575-079-1

CUPRINS

3 Cuprins
5 Introducere

9 CAP I. Prezentarea limbajului Turbo Pascal 6
37 CAP II. Grafica în Turbo Pascal

45 CAP III. Rezolvarea ecuațiilor
51 CAP IV. Derivarea și integrarea numerică
57 CAP V. Integrarea prin metoda cuadraturii Gauss
65 CAP VI. Integrarea ecuațiilor diferențiale ordinare
71 CAP VII. Rezolvarea ecuațiilor cu derivate parțiale prin metoda diferențelor finite
79 CAP VIII. Sisteme de ecuații liniare și neliniare
91 CAP IX. Statistica datelor experimentale
107 CAP X. Interpolarea exactă prin polinom
111 CAP XI. Interpolarea prin metoda celor mai mici pătrate
131 CAP XII. Găsirea rădăcinilor reale și complexe ale unui polinom
135 CAP XIII. Valori și vectori proprii
141 CAP XIV. Determinarea valorilor și vectorilor proprii ai unei matrici simetrice reale prin metoda Jacobi
151 CAP XV. Metoda Monte-Carlo
159 CAP XVI. Convoluția și deconvoluția spectrelor

171 Bibliografie

INTRODUCERE

Această lucrare are ca scop prezentarea și sistematizarea cunoștiințelor necesare pentru prelucrarea automată a datelor fizice , și se adresează în special studenților de la Facultatea de Fizică .

Cantitatea enormă , mereu crescândă , de rezultate experimentale pe care le furnizează diferitele tehnici experimentale , precum și complexitatea calculelor teoretice , impun cu necesitate prelucrarea acestora la calculatoare electronice .

In fizică se utilizează două tipuri de programe de calcul . Prima categorie constă din așa numitele programe teoretice de calcul , pentru evaluarea unor parametrii fizici în anumite ipoteze . Acești parametrii calculați sunt apoi comparați cu parametrii măsurați experimental . Din această comparație se trag concluzii cu privire la compatibilitatea ipotezelor teoretice cu realitatea practică . Pentru găsirea unei acceptabil de bune concordanțe e nevoie să se efectueze multe încercări , adică e nevoie de schimbarea de multe ori a parametrilor teoretici , iar adesea e nevoie de schimbarea însuși a modelului .

Al doilea tip de programe de calcul , programele de fit , sunt acelea în care în mod automat parametrii teoretici se potrivesc în așa fel încât să existe o concordanță cât mai bună cu datele experimentale care sunt introduse în calculator .

Necesitatea utilizării calculatorului în prelucrarea automată a datelor fizice , în vederea creșterii randamentului și eficienței cercetării științifice , precum și necesitatea ca aceste date să fie prelucrate corect din punct de vedere fizic și matematic determină nevoia perfecționării și modernizării procesului de educație a studenților și în această direcție .

Metodele numerice sunt expuse în ordinea dificultății și a complexității lor , cele simple , cele care stau la baza altor metode mai complexe , fiind expuse inițial . De aceea , pentru înțelegerea tuturor metodelor numerice prezentate este important ca această carte să fie studiată de la primul capitol , până la ultimul , cu continuitate , fără a se sări nici o parte . Prezentarea metodelor numerice , a algoritmilor , se face pe larg , se expune și teoria matematică implicată în metodă , nu numai pașii algoritmului , pentru ca întregul proces de calcul , de la introducerea datelor până la obținerea rezultatelor , să fie justificat din punct de

vedere matematic .

Deşi programele au fost scrise în Turbo Pascal 6 , ele pot fi uşor rescrise şi în alte limbaje , deoarece tratarea teoretică a metodelor numerice a fost făcută fără nici o referire la aplicaţia lor practică în limbajul sus amintit , algoritmii fiind expuşi într-o manieră cât mai generală .

Cunoştiinţele necesare prelucrării automate a datelor se dobândesc pe parcursul a trei etape :

1) Inţelegerea algoritmului
2) Stăpânirea limbajului de programare
3) Scrierea programului

Etapa a treia poate fi parcursă numai după ce au fost parcurse primele două .

Această lucrare nu-şi propune să fie un manual de Turbo Pascal 6 . Primul capitol face doar o scurtă recapitulare a lucrurilor esenţiale , pe care trebuie să la aibă în memorie orice programator în acest limbaj .

Un limbaj de programare nu se învaţă doar citind din cărţi , ci prin practică , stând zi de zi faţă în faţă cu calculatorul , scriind , testând , depanând şi rulând propriile programe . Experienţa a dovedit că studenţii care nu au exersat cu continuitate pe calculator , deşi Facultatea de Fizică este bine dotată în acest domeniu , nu au fost în stare să îşi scrie propriile programe , nereuşind nici să înţeleagă programele gata scrise .

După prezentarea teoretică a diferitelor metode numerice , de regulă urmează şi un program scris în Turbo Pascal 6 . Acest program este doar aplicaţia practică a metodei descrise înaintea lui , şi înţelegerea lui presupune cunoştiinţe temeinice de programare în limbajul Turbo Pascal 6 . Totuşi utilizarea acestor programe se poate face şi cu cunoştiinţe minimale de Turbo Pascal 6 , deoarece introducerea datelor iniţiale şi scoaterea datelor finale nu creează probleme deosebite . De multe ori programul utilizează şi grafica , deoarece vizualizarea unui rezultat ajută mult la înţelegerea lui , iar programul devine mai atractiv .

Capitolul al doilea tratează problema graficii în Turbo Pascal 6 . El trebuie studiat cu multă atenţie , pentru că toate programele ce folosesc grafica o fac prin funcţiile şi procedurile descrise aici .

Lucrarea conţine un mare număr de exemple , de probleme rezolvate şi de probleme propuse , menite să concretizeze noţiunile generale şi să formeze la studenţi o îndemânare în prelucrarea la calculator a datelor . Unele din problemele prezentate au fost preluate din ' Prelucrarea automată a datelor fizice ' de C.I.Nistor , din 1979 .

Toate programele au fost testate şi se află în

rețeaua de calculatoare a Facultății de Fizică , Universitatea București .

Mulțumiri deosebite aducem D-lui Prof.Dr. V.Grecu pentru materialul bibliografic pus la dispoziția noastră , precum și D-nei Conf.Dr. A.Constantinescu , pentru sugestiile utile făcute .

CAP I. PREZENTAREA LIMBAJULUI TURBO PASCAL 6

Pentru o rapidă învățare a limbajului de programare Turbo Pascal vă sugerăm următoarele etape :

1) Parcurgerea programului de instruire TPTOUR , care vine odată cu celelalte fișiere ale Turbo Pascal-ului . Astfel veți învăța să editați , să compilați , să depanați , să executați un program , și să utilizați HELP-ul . Toate aceste operațiuni în Turbo Pascal 6 se pot face cu ajutorul unui program interfață cu utilizatorul , numit IDE (Integrated Development Environment).

Compilarea unui program înseamnă trecerea lui din limbajul de programare în care a fost scris în cod mașină , pentru a putea fi executat de către calculator .

2) Citirea atentă a acestui capitol , studiind micile programe exemplificative .

3) Studierea tuturor cuvintelor limbajului Pascal ce apar în acest capitol , cu ajutorul HELP-ului , urmărind cu atenție și exemplele date în HELP . Toate aceste cuvinte au fost scrise cu majuscule în acest capitol .

Compilatorul Pascal nu face diferența între literele majuscule și cele minuscule .

1) Tipuri de date

Funcția principală a calculatorului este prelucrarea datelor .

Fiecare limbaj de programare are propriile lui tipuri de date .

In Pascal orice variabilă trebuie asociată explicit , printr-o declarație , cu un anumit tip de dată .

Limbajul Turbo Pascal 6 are următoarele tipuri de date :

1) tipuri ordinale (ORDINAL)

Aceste tipuri reprezintă mulțimi finite de valori , între oricare două elemente existând o relație de ordine de tipul 'mai mic' sau 'mai mare' . Tipurile ordinale sunt :

BOOLEAN
CHAR
SHORTINT , INTEGER , LONGINT , BYTE , WORD
SET
ENUMERATED
SUBRANGE

Variabilele tipului BOOLEAN sunt doar două , TRUE și FALSE .

Variabilele de tip CHAR sunt literele

alfabetului , cifrele , alte simboluri .
Variabilele de tip CHAR se scriu între ghilimele : 'x'

Variabilele de tipurile SHORTINT , INTEGER , LONGINT , BYTE , WORD sunt numere întregi , fiecare tip având o valoare minimă și o valoare maximă specifice .

Variabilele de tipul SET și ENUMERATED sunt definite de utilizator , după anumite regului .

Variabilele de tipul SUBRANGE provin dintr-un tip ordinal general , prin limitarea la un subdomeniu .

2) tipul STRING , șir de caractere

Variabilele de acest tip se scriu între ghilimele : 'abc'

Atenție ! Caracterul 'x' și string-ul 'x' nu sunt echivalente .

3) tipuri reale

Tipurile reale sunt :

REAL , SINGLE , DOUBLE , EXTENDED , COMP

Tipul COMP de fapt conține numai valori întregi , dar în calcule intră ca un număr real fără parte zecimală .

Notația este : $xEy = x \cdot 10^y$, cu $-10 < x < 10$

4) tipuri structurate

ARRAY

RECORD

Aceste tipuri sunt definite de utilizator , în funcție de variabilele necesare în program .

Exemple :

```
array[1..10,1..5] of integer;
record i:integer; r:real; end;
```

5) tipul POINTER , folosit pentru adresarea în memorie a variabilelor dinamice (cele ce nu sunt adresate prin nume , ci prin adresa zonei de memorie în care se află) .

6) tipul PROCEDURE și tipul FUNCTION , folosite în cazul în care printre parametrii unei proceduri sau funcții se află chiar o procedură sau o funcție . In acest caz programul trebuie compilat cu directiva {SF+} .

7) tipul fișier

Acest tip admite trei forme :

FILE (fișier de date oarecare)

FILE OF ... (fișier de date de un anume tip)

TEXT (fișier de date ce conține linii de text , sfârșitul fiecărei linii fiind marcat cu un caracter special)

Variabilele de tip fișier sunt puse în corespondență cu denumirile fișierelor ce sunt memorate pe suportul magnetic (floppy-disk sau hard-disk) .

2) Operatori

1) Operațiile ce se pot face cu numere întregi sunt :

+ adunarea
- scăderea
* înmulţirea
DIV împărţirea întreagă (câtul împărţirii)
MOD clase de resturi modulo (restul împărţirii)

2) Operaţiile ce se pot face cu numere reale sunt :
+ adunarea
- scăderea
* înmulţirea
/ împărţire reală

3) Operaţiile ce se pot face cu date logice (BOOLEAN) sunt :
AND şi logic
OR sau logic
XOR sau exclusiv logic
NOT negaţie logică

3) Relatiile de ordine

Intre două date ale unui tip ordinal pot avea loc următoarele relaţii de ordine :

```
<   mai mic
<=  mai mic sau egal
>   mai mare
>=  mai mare sau egal
=   egal
<>  diferit
```

4) Conversii de tipuri

Intre diferitele tipuri de date se pot face conversii cu ajutorul următoarelor funcţii sau proceduri :
VAL transformă un string într-o valoare numerică
STR transformă o valoare numerică într-un string
TRUNC trunchează un număr real la o valoare întreagă
ROUND rotunjeşte un număr real la o valoare întreagă

5) Structura generala a unui program

Un program scris în Turbo Pascal 6 are următoarea structură :

```
PROGRAM numele_programului;
USES ... ;
LABEL ... ;
CONST ... ;
TYPE ... ;
VAR ... ;
PROCEDURE ... ;
FUNCTION ... ;
BEGIN
comandă;
```

```
... ;
END.
```

De regulă comenzile sunt separate prin caracterul ';' .

Comanda PROGRAM este urmată de numele programului . Acest nume nu trebuie să conțină decât un cuvânt , nu este permisă folosirea în el a spațiilor albe . Exemplu :

```
PROGRAM rezolvare_ecuatie;
```

Comanda USES este urmată de numele UNIT-urilor folosite de programul principal . Un UNIT este o librărie de funcții și proceduri , de declarații de tipuri de date , de constante , de variabile . Vom avea două tipuri de UNIT-uri : cele ce vin odată cu compilatorul , și cele făcute de utilizatori . Acestea din urmă trebuie să fie compilate , și să se găsească ori în directorul de lucru , ori în cel specificat în Option|Directories|Unit Directories . Prin compilarea pe disc un UNIT dă naștere la un fișier cu extensia .TPU (Turbo Pascal Unit) .

UNIT-urile ce vin odată cu compilatorul sunt: CRT , DOS , GRAPH , OVERLAY , PRINTER , SYSTEM . Exemplu :

```
USES crt;
```

Trebuie precizat că orice program are acces la UNIT-ul SYSTEM , chiar dacă acesta nu este declarat în USES .

Comanda LABEL specifică niște etichete , folosite de către comenzile GOTO . Exemplu :

```
LABEL abc;
```

Comanda :

```
GOTO abc;
```

obligă programul să execute un salt până în locul în care întâlnește eticheta :

```
abc:
```

Comanda CONST definește niște constante , sau niște variabile inițializate . Pentru constante avem comanda :

```
CONST nume_constanta=valoare;
```

Exemplu :

```
CONST i=100;
```

Constantele nu își pot modifica valoarea în timpul executării programului .

Pentru variabile inițializate avem comanda :

```
CONST nume_variabila:tip_variabila=valoare;
```

unde tip_variabila este tipul căruia îi aparține variabila .

Exemplu :

```
CONST i:integer=100;
```

Variabilele inițializate își pot modifica valoarea în timpul executării programului .

Comanda TYPE definește noi tipuri de variabile . Cel mai frecvent tip de variabilă definit de utilizator este matricea . De exemplu , spre a defini un tip de variabilă căruia să-i aparțină

matricile pătratice de dimensiune 10 , cu elemente reale , vom folosi comanda :

```
TYPE matrice=array[1..10,1..10] of real;
```

Comanda generală are forma :

```
TYPE tip_variabila= ... ;
```

Definirea de noi tipuri de variabile apare ca o necesitate când variabile de acest fel apar ca parametrii într-o procedură sau funcție .

Comanda VAR introduce variabilele globale . Aceste variabile , ca și cele introduse la constantele inițializate , sunt accesibile din orice parte a programului principal , precum și din procedurile și funcțiile lui . Comanda generală este de felul :

```
VAR nume_variabila:tip_variabila;
```

Exemplu :

```
VAR i:integer; a:matrice;
```

După aceste instrucțiuni , dintre care nici una nu este obligatoriu de folosit , pot apare proceduri sau funcții , folosite de către programul principal sau de către alte proceduri sau funcții .

O procedură este de tipul :

```
PROCEDURE nume_proc(nume_var:tip_var ; ... );
   VAR ... ;
   BEGIN
   comandă;
   ... ;
   END;
```

Deoarece suntem obligați să specificăm și tipul variabilelor date ca parametri , tipurile nestandard vor trebui declarate cu ajutorul instrucțiunii TYPE .

Dacă variabilele date ca parametrii pot fi modificate de către procedură , ele vor fi precedate în declarația PROCEDURE de către cuvântul VAR . Acesta este un mod prin care o procedură poate returna niște rezultate , celelalte moduri fiind : prin accesul la variabilele globale , prin scrierea pe monitor sau prin tipărirea la imprimantă .

Variabilele declarate de către comanda PROCEDURE , și utilizate în interiorul procedurii , poartă numele de parametrii formali. In momentul când procedura este executată , în locul acestor parametrii se vor pune niște variabile sau constante de același tip , numite parametrii actuali , pentru care se va executa programul procedurii .

Instrucțiunea VAR ce urmează declarației procedurii introduce variabilele locale . Ele pot fi accesate numai din interiorul respectivei proceduri . In cazul în care un nume de variabilă este folosit și la declararea unei variabile globale , și la declararea unei variabile locale,

atunci în interiorul procedurii numele variabilei va avea semnificaţia variabilei locale .
O funcţie este de tipul :

```
FUNCTION nume_func(nume_var:tip_var ; ... ):tip_;
    VAR ... ;
    BEGIN
    comandă;
    ... ;
    nume_func:= ... ;
    ... ;
    END;
```

Sunt valabile toate observaţiile de la proceduri , iar în plus mai avem de discutat rezultatul pe care îl returnează funcţia . Acesta trebuie să fie de tipul tip_ , cu precizarea că nu orice tip de variabilă poate fi folosit pentru a caracteriza rezultatul unei funcţii . Tipul tip_ poate fi ORDINAL, REAL, STRING sau POINTER .
In cadrul funcţiei trebuie să apară cel puţin într-un loc o instrucţiune de atribuire , prin care numelui funcţiei i se dă o valoare , de tipul tip_ .

6) Directivele de compilare

Porţiuni de program , sau tot programul , pot fi compilate cu diferite metode . Directivele de compilare se scriu între acolade , prima literă fiind '$' , şi au efect asupra porţiunii de program ce le urmează . Cele mai folosite directive de compilare sunt :
{$F+} folosită pentru ca apelul procedurilor şi al funcţiilor să conţină adresa completă , şi nu doar adresa relativă . Această directivă de compilare permite ca o funcţie sau o procedură să poată să fie dată ca parametru unei alte funcţii sau proceduri .
{$N+,E+} folosită pentru emularea coprocesorului matematic , pentru a executa calcule cu o precizie sporită , cu variabile aparţinând unor tipuri mai exacte . Acestă directivă nu trebuie folosită în cazul în care calculatorul e dotat cu coprocesor matematic .

7) Instructiuni conditionale si de ciclare

O caracteristică a limbajului de programare TP este apariţia grupurilor ' BEGIN ... END ' . Acestea delimitează corpul programului principal, corpul procedurilor şi funcţiilor , dar pot delimita şi grupuri de comenzi care trebuiesc executate în bloc , ca o singură comandă , notate <comandă> . Aceste grupuri de comenzi apar cel mai adesea în instrucţiuni de ciclare sau în ramificaţii ce urmează testării unei condiţii .

Aceste instrucţiuni sunt :

```
FOR contor:=v1 TO v2 DO <comandă>;
                              ciclare necondiţionată
FOR contor:=v1 DOWNTO v2 DO <comandă>;
                              ciclare necondiţionată
REPEAT <comandă> UNTIL condiţie;
                    ciclare condiţionată , cu testarea
                 finală a condiţiei de ieşire din ciclu
```

Expresia sau variabila condiţie trebuie să fie de tipul BOOLEAN .

In această instrucţiune , în cazul unui bloc de comenzi , identificatorii BEGIN şi END nu mai trebuie puşi , rolul lor fiind luat de REPEAT şi UNTIL .

```
WHILE condiţie DO <comandă>;
                    ciclare condiţionată , cu testarea
               iniţială a condiţiei de ieşire din ciclu
IF condiţie THEN <comandă>;
                              ramificaţie logică IF
IF condiţie THEN <comandă> ELSE <comandă>;
                              ramificaţie logică IF
CASE variabilă OF      ramificaţie logică CASE
val1: <comandă>;
val2: <comandă>;
END;
CASE variabilă OF      ramificaţie logică CASE
val1: <comandă>;
val2: <comandă>;
ELSE <comandă>
END;
```

Variabila folosită de instrucţiunea CASE trebuie să fie de un tip ORDINAL .

8) Structura generala a unui unit

Un unit scris în Turbo Pascal 6 are următoarea structură :

```
   UNIT nume_unit;
INTERFACE
   USES ... ;
   CONST ... ;
   TYPE ... ;
   VAR ... ;
   PROCEDURE ... ;
   FUNCTION ... ;
IMPLEMENTATION
   USES ... ;
   LABEL ... ;
   CONST ... ;
   TYPE ... ;
   VAR ... ;
   PROCEDURE ... ;
   FUNCTION ... ;
END.
```

In secţiunea INTERFACE sunt declarate

constante , tipuri de variabile , variabile , precum și denumirile procedurilor și funcțiilor ce pot fi folosite de către utilizatorul unit-ului . Corpul acestor proceduri și funcții va fi scris doar în secțiunea IMPLEMENTATION . Aici nu mai suntem obligați ca după denumirea procedurilor și funcțiilor respective să mai scriem și parametrii formali . Secțiunea IMPLEMENTATION mai poate conține și alte declarații de constante , tipuri de variabile , variabile , proceduri și funcții altele decât cele menționate în secțiunea INTERFACE , dar acestea nu vor fi accesibile utilizatorilor unit-ului , ci vor putea fi folosite doar de către proceduri și funcții ale însăși unit-ului .

9) Programe exemplificative

demo1.pas

Cel mai simplu program scris în Pascal este programul care nu face nimic . El conține doar instrucțiunile BEGIN și END. , ce delimitează corpul programului principal . Opțional se poate da și un nume programului , cu ajutorul instrucțiunii PROGRAM . Orice comentariu făcut de programator pentru a-l ajuta să înțeleagă mai bine programul , și care nu este o instrucțiune ce trebuie compilată , va fi pus între două acolade { } .
Iată programul :

```
PROGRAM demo1;
{acest program nu face nimic}
begin
end.
```

demo2.pas

Scrierea unor date pe monitor se poate face cu ajutorul comenzilor WRITE și WRITELN . Comanda WRITE scrie data la poziția cursorului și deplasează cursorul după data scrisă , pe același rând . Comanda WRITELN scrie data la poziția cursorului și deplasează cursorul la începutul rândului următor . Rândurile pe monitor se ocupă de sus în jos . Programul următor scrie niște constante de tipul CHAR , dar la fel de bine puteau fi și de alt tip , sau variabile . Comanda READLN de la sfârșit are rolul de a introduce o pauză , pentru menținerea informației pe monitor, până când se apasă tasta <ENTER> .
Iată programul :

```
PROGRAM demo2;
begin
    write('a');
```

```
    write('b');
    writeln('c');
    write('d');
    writeln('e');
    readln;
end.
```

In locul primelor trei instrucţiuni de scriere puteam la fel de bine să introducem comanda :

writeln('abc');

sau comanda :

writeln('a','b','c');

comenzile fiind echivalente cu cele trei instrucţiuni .

<u>demo3.pas</u>

Citirea datelor de la tastatură se face cu comenzile READ şi READLN , ambele scriind pe monitor , la poziţia cursorului , data ce este introdusă de la tastatură , iar după apăsarea tastei <ENTER> , ce semnifică sfârşitul introducerii datelor , cursorul este deplasat la începutul liniei următoare . Diferenţierea între cele două comenzi se face când acestea citesc date dintr-un fişier , comanda folosită depinzând de tipul fişierului de date .

Variabila folosită în program este de tipul REAL .

După cum s-a mai discutat , comanda READLN ce nu are nici o variabilă ca parametru are funcţia de a introduce o pauză , până la apăsarea tastei <ENTER> .

In comanda de scriere , aşa cum se vede din exemplu , pot fi introduse pentru scriere , pe lângă constante şi variabile , şi expresii , acestea fiind întâi calculate , iar apoi rezultatul este scris pe monitor .

Iată programul :

```
PROGRAM demo3;
var x:real;
begin
    write('x = ');
    readln(x);
    writeln('x*x = ',sqr(x));
    readln;
end.
```

Funcţia SQR (square) efectuează ridicarea la pătrat . La fel de bine în locul ei puteam introduce următoarea expresie , echivalentă :

writeln('x*x = ',x*x);

Funcţia SQRT (square root) efectuează extragerea rădăcinii pătrate .

demo4.pas

Acest program este foarte asemănător cu precedentul , numai că funcția ce o calculează nu este predefinită de către limbajul Pascal , ci este definită de către utilizator .

Funcția definită este ridicarea la putere , cu exponent real . Se folosește următoarea formulă :

$$x^y = e^{y \cdot \ln(x)}$$

ce utilizează funcțiile EXP și LN , funcții predefinite ale limbajului Pascal , ce calculează exponențiala , respectiv logaritmul unui număr real .

Parametrii formali , declarați de către instrucțiunea FUNCTION , sunt baza și putere . Când funcția este executată locul acestor parametrii este luat de parametrii actuali , în cazul nostru x și y . Pentru a face ușoară distincția între parametrii formali și cei actuali , ei au fost denumiți cu nume diferite . Totuși , în unele programe , pentru a ușura modul în care este înțeleasă acțiunea concretă a funcției sau procedurii de prelucrare a parametrilor actuali , se va prefera ca parametrii formali și cei actuali să aibă același nume . In acest caz nu trebuie uitat că funcția sau procedura poate modifica doar parametrii globali (definiți de instrucțiunea VAR a programului principal sau a unit-urilor folosite de acestea) , sau parametrii actuali ce iau în funcție sau procedură locul unor parametrii formali precedați de cuvântul rezervat VAR .

Iată programul :

```
PROGRAM demo4;
var x,y:real;

FUNCTION putere(baza,exponent:real):real;
begin
     putere:=exp(exponent*ln(baza));
end;

begin
     writeln('Programul calculeaza pe x^y');
     writeln('x,y sunt numere reale si x>0');
     write('x = ');
     readln(x);
     write('y = ');
     readln(y);
     writeln('x^y = ',putere(x,y));
     readln;
end.
```

demo5.pas

Acest program este asemănător cu precedentul ,

doar că funcția calculată este ridicarea la putere , cu exponent întreg .

Variabilele folosite sunt de tipul REAL și INTEGER .

Funcția folosită de data aceasta mai introduce două variabile , prin instrucțiunea VAR , variabilele i și s , ce au existență doar în cadrul funcției .

Astfel , variabilele locale sunt i și s , variabilele globale fiind x și n .

Calculul ridicării la putere se face prin înmulțiri repetate , pentru exponent pozitiv , și prin împărțiri repetate,pentru exponent negativ .

Ramificația în program rezultată în urma existenței celor două posibilități , ca semnul exponentului să fie pozitiv sau negativ , este rezolvată cu ajutorul instrucțiunii IF . In acest mod este păstrat rezultatul corect și pentru cazul în care exponentul este 0 , funcția returnând în acestă situație valoarea 1 .

Inmulțirea repetată , respectiv împărțirea repetată , sunt efectuate cu ajutorul unor instrucțiuni de ciclare .

Instrucțiunea

FOR i:=i1 TO i2 DO <comandă>;

efectuează comanda pentru fiecare valoare a lui i , incepând cu i1 și terminând cu i2 , după fiecare efectuare a comenzii variabila i fiind mărită cu o unitate . Deci , pentru efectuarea măcar o dată a comenzii , este necesar ca i1≤i2 . In cazul în care i1>i2 programul trece la instrucțiunea următoare , neefectuând comanda din ciclu nici măcar o dată .

Instrucțiunea

FOR i:=i1 DOWNTO i2 DO <comandă>;

efectuează comanda pentru fiecare valoare a lui i , începând cu i1 și terminând cu i2 , după fiecare efectuare a comenzii variabila i fiind micșorată cu o unitate . Deci , pentru efectuarea măcar o dată a comenzii , este necesar ca i1≥i2 . In cazul în care i1<i2 programul trece la instrucțiunea următoare , neefectuând comanda din ciclu nici măcar o dată .

Iată programul :

```
PROGRAM demo5;
var x:real;
    n:integer;

FUNCTION PUTERE(baza:real;exponent:integer):real;
var i:integer; s:real;
begin
     s:=1;
     if exponent<0
     then for i:=-1 downto exponent do s:=s/baza;
     if exponent>0
     then for i:=1 to exponent do s:=s*baza;
```

```
    putere:=s;
end;

begin
    writeln('Programul calculeaza pe x^n');
    writeln('n este numar intreg');
    write('x = ');
    readln(x);
    write('n = ');
    readln(n);
    writeln('x^n = ',putere(x,n));
    readln;
end.
```

demo6.pas

Dacă ați înțeles programele precedente , programul următor îl veți putea urmări foarte ușor . Funcția calculată aici este factorialul unui număr natural , iar singura noutate este apariția tipului de date LONGINT , pentru a lucra cu numere întregi foarte mari . Acest lucru este necesar deoarece factorialele , chiar ale unor numere mici , au valori foarte mari .

Factorialul este calculat cu ajutorul unei înmulțiri repetate , în ciclul deschis prin instrucțiunea FOR ... DO ... .

Pentru a limita numărul căruia îi calculăm factorialul la valori suficient de mici astfel încât factorialul său să nu depășească valoarea maximă a tipului LONGINT , numărul a fost declarat ca aparținând unui tip de variabilă SUBRANGE , dat prin intervalul 1..12 . Astfel numărul este obligat să aparțină acestui interval .

Iată programul :

```
PROGRAM demo6;
var n:1..12;
   m:longint;

FUNCTION fact(n:integer):longint;
var i:integer;
   p:longint;
begin
     p:=1;
     for i:=2 to n do p:=p*i;
     fact:=p;
end;

begin
     writeln('Programul calculeaza n!');
     writeln('n este cuprins intre 1 si 12');
     write('n = ');
     readln(n);
     m:=fact(n);
     writeln('n! = ',m);
```

```
   readln;
end.
```

Modificați programul , astfel încât să puteți introduce pentru n și valori mai mari decât 12 . Ce observați ? Rezultatul poate fi un număr negativ , datorită depășirii valorii maxime permise variabilelor de tipul LONGINT .

O sursă importantă de erori este neprotejarea programului în fața introducerii , de la tastatură , de obicei , a unor date nepotrivite , cărora programul n-ar trebui să li se aplice .

Un program corect scris trebuie ori să dea rezultatul corect , ori să dea un mesaj de eroare dacă dintr-un motiv sau altul calculul trebuie să se oprească . Programul nu trebuie ca în unele cazuri să dea rezultatul corect (în exemplul nostru , pentru n:=1..12) , iar în altele un rezultat eronat (în exemplul nostru , pentru n>12) .

<u>demo7.pas</u>

Acest program află elementul maxim al unui vector (o matrice cu o singură dimensiune) . Se definesc două tipuri de date : tipul dec , ce este un tip SUBRANGE , și tipul mc (de la Matrice Coloană) , ce este un tip ARRAY . Intervalul pe care îl poate lua indicele elementelor matricii este dat între parantezele drepte . In cazul în care indicele poate lua toate valorile unui tip de date ORDINAL , atunci între parantezele drepte poate fi scris chiar acel tip . Pentru cazul general când doar un interval anume este destinat indicelui , el va fi dat sub forma i..j în interiorul parantezelor drepte . Pentru cazul nostru , o declarație echivalentă a tipului de date mc este :

```
type mc=array[1..10] of real;
```

Introducerea tipului de date dec este utilă deoarece limitează implicit valoarea dimensiunii vectorului , altfel , lucrând cu date de tip INTEGER , ar fi trebuit să introducem suplimentar o instrucțiune IF pentru a verifica dacă utilizatorul programului a introdus pentru n o valoare ce într-adevăr aparține intervalului [1,10] . O altă observație importantă este că e bine ca întotdeauna când solicităm de la tastatură o dată ce nu aparține unui tip standard (REAL , INTEGER , CHAR , STRING) să atenționăm și asupra valorilor ce le poate lua data introdusă de la tastatură .

Introducerea tipului de date mc este absolut necesară , deoarece o variabilă de acest tip apare ca parametru formal al funcției , în instrucțiunea FUNCTION .

Urmăriți cu atenție ciclul din programul

principal , prin care se introduc valorile elementelor vectorului . Această structură o vom întâlni ori de câte ori vom introduce date într-o matrice .

Selecția elementului maxim al vectorului se face în ciclul din corpul funcției , unde rolul cheie îl joacă instrucțiunea IF .

Iată programul :

```
PROGRAM demo7;
type dec=1..10;
     mc=array[dec] of real;
var x:mc;
    n,i:dec;
    max:real;

FUNCTION maxim(n:dec; x:mc):real;
var i:dec;
    m:real;
begin
     m:=x[1];
     for i:=2 to n do
        if m<x[i] then m:=x[i];
     maxim:=m;
end;

begin
    writeln('Aflarea elem maxim al unui vector');
    writeln('Introduceti dim vectorului(1..10)');
    write('n = ');
    readln(n);
    writeln('Introduceti elementele vectorului');
    for i:=1 to n do
    begin
         write('x[',i,'] = ');
         readln(x[i]);
    end;
    max:=maxim(n,x);
    writeln('Elem maxim al vectorului este');
    writeln('max = ',max);
    readln;
end.
```

<u>demo8.pas</u>

Acest program face același lucru ca și precedentul , numai că selecția elementului maxim din vector nu se face cu o funcție , ci cu o procedură .

Pentru ca procedura să poată să transmită rezultatul prin intermediului parametrului formal max , acesta este precedat de cuvântul rezervat VAR , în declarația PROCEDURE .

Parametrii actuali care corespund unor parametri formali ce nu sunt precedați de cuvântul VAR în declarația FUNCTION sau PROCEDURE nu sunt modificați ca urmare a executării

respectivei funcții sau proceduri .
Parametrii actuali care corespund unor parametri formali ce sunt precedați de cuvântul VAR în declarația FUNCTION sau PROCEDURE sunt modificați ca urmare a executării respectivei funcții sau proceduri . Ei vor lua ultima valoare pe care au avut-o parametrii formali atașați lor, înainte ca programul să iasă din respectiva funcție sau procedură .
Iată programul :

```
PROGRAM demo8;
type dec=1..10;
     mc=array[dec] of real;
var x:mc;
    n,i:dec;
    max:real;

PROCEDURE maxim(n:dec; x:mc; var max:real);
var i:dec;
begin
     max:=x[1];
     for i:=2 to n do
        if max<x[i] then max:=x[i];
end;

begin
    writeln('Aflarea elem maxim al unui vector');
    writeln('Introduceti dim vectorului(1..10)');
    write('n = ');
    readln(n);
    writeln('Introduceti elementele vectorului');
    for i:=1 to n do
    begin
         write('x[',i,'] = ');
         readln(x[i]);
    end;
    maxim(n,x,max);
    writeln('Elem maxim al vectorului este');
    writeln('max = ',max);
    readln;
end.
```

O altă posibilitate de a transmite rezultatul unei proceduri este de a folosi o variabilă globală , ce poate fi modificată și din interiorul procedurilor și funcțiilor .
Astfel , declarația PROCEDURE ar fi putut să fie :

```
PROCEDURE maxim(n:dec; x:mc);
```

în acest caz apelul procedurii trebuind să fie :

```
maxim(n,x);
```

Variabila max din interiorul procedurii , ce în programul inițial era o variabilă locală , în cazul modificărilor descrise mai sus , datorită declarației VAR din cadrul programului principal, devine variabilă globală (și în interiorul

procedurii) .

demo9.pas

Programul următor are ca scop să introducă un tip mai complex de dată , și anume o matrice cu două dimensiuni (cu linii și coloane) . Familiarizați-vă cu modul de introducere a elementelor matricii , prin dublul ciclu după i și j . Un ciclu dublu asemănător este folosit pentru identificarea elementului maxim al matricii . Această identificare este făcută în cadrul programului principal , nu a mai fost scrisă sub formă de funcție sau procedură , pentru că scopul acestui program nu este să vă familiarizeze cu utilizarea funcțiilor și procedurilor , ci să vă familiarizeze cu lucrul cu matrici , cu folosirea ciclurilor duble . Puteți încerca, pe post de exercițiu, modificarea programului , astfel încât selecția elementului maxim să fie făcută de o funcție sau de o procedură .

Iată programul :

```
PROGRAM demo9;
var a:array[1..10,1..10] of real;
    n,m,i,j:1..10;
    max:real;
begin
   writeln('Aflarea elem maxim al unei matrici');
   writeln('Introd nr de linii ale mat (1..10)');
   write('n = ');
   readln(n);
   writeln('Introd nr de col ale mat (1..10)');
   write('m = ');
   readln(m);
   writeln('Introduceti elementele matricii');
   for i:=1 to n do
   for j:=1 to m do
   begin
        write('a[',i,',',j,'] = ');
        readln(a[i,j]);
   end;
   max:=a[1,1];
   for i:=1 to n do
   for j:=1 to m do
   if max<a[i,j] then max:=a[i,j];
   writeln('Elementul maxim al matricii este');
   writeln('max = ',max);
   readln;
end.
```

demo10.pas

Următorul program calculează urmele a două matrici pătratice , și apoi calculează matricea ce rezultă din înmulțirea celor două matrici .

Reamintim că trebuie definit un tip de dată mat deoarece variabile de acest tip apar ca parametrii formali în declarațiile PROCEDURE și FUNCTION .

Urmăriți cu atenție ciclul triplu , după i , j și k , din interiorul procedurii prod_mat .

Rezultatul procedurii este transmis prin intermediul variabilei c , parametru formal precedat de cuvântul VAR în declarația PROCEDURE, astfel încât variabila globală c , devenită parametru actual prin comanda prod_mat , după terminarea execuției procedurii , va lua ultima valoare a parametrului formal c .

Iată programul :

```
PROGRAM demo10;
type mat=array[1..10,1..10] of real;
var n,i,j:1..10;
    a,b,c:mat;

PROCEDURE prod_mat(a,b:mat;n:integer; var c:mat);
var i,j,k:integer; s:real;
begin
     for i:=1 to n do
     for j:=1 to n do begin
        s:=0;
        for k:=1 to n do s:=s+a[i,k]*b[k,j];
        c[i,j]:=s; end;
end;

FUNCTION trace(a:mat; n:integer):real;
var i:integer; s:real;
begin
     s:=0;
     for i:=1 to n do s:=s+a[i,i];
     trace:=s;
end;

begin
     writeln('Programul efectueaza inmultirea');
     writeln('a doua matrici patratice');
     writeln('si le calculeaza urmele');
     writeln('Introduceti ordinul mat (1..10)');
     write('n = ');
     readln(n);
     writeln('Introduceti elem primei matrice');
     for i:=1 to n do
     for j:=1 to n do
     begin
          write('a[',i,',',j,'] = ');
          readln(a[i,j]);
     end;
     writeln('Introd elem celei de-a doua mat');
     for i:=1 to n do
     for j:=1 to n do
     begin
          write('b[',i,',',j,'] = ');
```

```
            readln(b[i,j]);
        end;
        prod_mat(a,b,n,c);
        writeln('Urmele matricilor sunt');
        writeln('trace(a) = ',trace(a,n));
        writeln('trace(b) = ',trace(b,n));
        writeln('Matricea produs c=ab este');
        writeln('(<ENTER> dupa fiecare linie)');
        for i:=1 to n do
        begin
            for j:=1 to n do
                writeln('c[',i,',',j,'] = ',c[i,j]);
            readln;
        end;
end.
```

Scrieți un program care să efectueze produsul a două matrici dreptunghiulare , după o prealabilă verificare a condiției ca numărul de coloane al celei dintâi să fie egal cu numărul de linii al celei de-a doua .

demo11.pas

Programul următor , deși este destul de lung , nu introduce ca elemente noi decât o variabilă de tipul RECORD , care este utilizată pe post de număr complex , cele două câmpuri ale ei fiind rezervate părții reale și părții imaginare ale numărului complex respectiv .

Programul calculează valoarea și modulul unui polinom cu coeficienți reali și de variabilă complexă .

Valoarea polinomului este calculată prin sumarea tuturor puterilor , fiecare termen aducând o contrbuție la partea reală și la cea imaginară ale polinomului . Se sumează începând cu termenul liber , și terminând cu termenul corespunzător puterii maxime .

Partea reală și cea imaginară ale lui z^i sunt calculate pornind de la partea reală și cea imaginară ale lui z^{i-1} . In felul acesta , prin ciclul FOR ... DO ... din procedura val_polinom , se calculează și se sumează toți termenii polinomului .

Iată programul :

```
PROGRAM demo11;
type grad=0..10;
     vector=array[grad] of real;
     { matrice pt coeficienti polinom }
     complex=record            { partea reala si }
               re,im:real;     { imaginara a unui }
             end;              { nr complex }
var c:vector;
    n,i:grad;
    a,b:real;
```

```
    s:complex;

PROCEDURE val_polinom(c:vector; n:grad;
                      a,b:real; var s:complex);
var i:grad; pr,pi,zr,zi,zrn,zin:real;
begin
     pr:=0; {val initiala a partii reale a polin}
     pi:=0; {val initiala a partii imag a polin}
     {la aceste valori se vor suma contributiile
     tuturor coeficientilor polinomului c[i]*z^i}
     zr:=1; {partea reala a lui z^0}      {z^0 = 1}
     zi:=0; {partea imaginara a lui z^0}
     {in cadrul ciclului ce urmeaza aceste
     variabile vor contine partea reala si partea
     imaginara ale lui z^i}
     for i:=0 to n do
     {se aduna contributiile termenilor z^i}
     begin
         pr:=pr+zr*c[i]; {realul polinomului}
         pi:=pi+zi*c[i]; {imaginarul polinomului}
         zrn:=zr*a-zi*b;
         {realul lui z^(i+1) , pornind de la z^i}
         zin:=zr*b+zi*a;
         {imag lui z^(i+1) , pornind de la z^i}
         zr:=zrn; {se trece la puterea urmatoare}
         zi:=zin; {a lui z}
     end;
     s.re:=pr;
     s.im:=pi;
end;

begin
     writeln('Acest program calculeaza ',
             'valoarea si modulul unui');
     writeln('polinom cu coeficienti reali ',
             'si de variabila complexa');
     writeln('de forma :');
     writeln('P(z) = ',
             'c[n]*z^n + ... + c[1]*z + c[0]');
     writeln('Introduceti gradul polin (0..10)');
     write('n = ');
     readln(n);
     writeln('Introduceti coeficientii polin');
     for i:=n downto 0 do
     begin
          write('c[',i,'] = ');
          readln(c[i]);
     end;
     writeln('Introd pct in care se calc polin');
     writeln('z = Re(z) + i*Im(z)');
     write('Re(z) = ');
     readln(a);
     write('Im(z) = ');
     readln(b);
     writeln('P(z) = Re(P(z)) + i*Im(P(z))');
     val_polinom(c,n,a,b,s);
     writeln('Re(P(z)) = ',s.re);
```

```
    writeln('Im(P(z)) = ',s.im);
    writeln('Modulul polinomului este');
    writeln('|P(z)| = ',
            sqrt(s.re*s.re+s.im*s.im));
    readln;
end.
```

demo12.pas

Acest program efectuează împărțirea unui polinom la un monom de forma (x + a) , și la un binom de forma (x^2 + a·x + b) , în ambele cazuri returnându-se doar polinomul cât , nu și restul .

Prin împărțirea la un monom , gradul polinomului scade cu 1 , iar prin împărțirea la un binom , gradul polinomului scade cu 2 . De acest lucru programul ține seama , și aici este singura subtilitate de calcul .

Iată programul :

```
PROGRAM demo12;
type grad=0..10;
     vector=array[grad] of real;
     { matrice pt coeficienti polinom }
var n,i:grad;
    p,c:vector;
    a,b:real;

procedure imp_monom(p:vector; var c:vector;
                    n:grad; a:real);
var i:grad;
begin { impartire la monomul (x + a) }
      for i:=n-1 downto 0 do
      begin
           c[i]:=p[i+1];
           p[i]:=p[i]-c[i]*a;
           { in final p[0]=restul }
      end;
end;

procedure imp_binom(p:vector; var c:vector;
                    n:grad; a,b:real);
var i:grad;
begin { impartire la binomul (x^2 + a*x + b }
      for i:=n-2 downto 0 do
      begin
            c[i]:=p[i+2];
            p[i+1]:=p[i+1]-c[i]*a;
            p[i]:=p[i]-c[i]*b;
            { in final p[1]*x + p[0]=restul }
      end;
end;

begin
     writeln('Acest program efectueaza ',
             'impartirea unui polinom');
     writeln('la un monom si la un binom . ',
```

```
          'Polinomul are forma :');
     writeln('P(z) = ',
          'p[n]*z^n + ... + p[1]*z + p[0]');
     writeln('Introduceti gradul polin (0..10)');
     write('n = ');
     readln(n);
     writeln('Introduceti coeficientii polin');
     for i:=n downto 0 do
     begin
          write('p[',i,'] = ');
          readln(p[i]);
     end;
     writeln('Introduceti monomul : (x + a)');
     write('a = ');
     readln(a);
     imp_monom(p,c,n,a);
     writeln('Coeficientii polin cit sunt');
     for i:=n-1 downto 0 do
        writeln('c[',i,'] = ',c[i]);
     writeln('Introd binomul (x^2 + a*x + b)');
     write('a = ');
     readln(a);
     write('b = ');
     readln(b);
     imp_binom(p,c,n,a,b);
     writeln('Coeficientii polin cit sunt');
     for i:=n-2 downto 0 do
        writeln('c[',i,'] = ',c[i]);
     readln;
end.
```

Modificați programul , astfel încât să se afișeze și polinomul rest , pentru cele două cazuri (monom și binom) . Vor trebui introduși niște parametrii formali suplimentari în proceduri .

canta.pas

Acest program are rolul de a arăta cum pot fi folosiți parametrii din linia de comandă . Acești parametrii sunt toate cuvintele ce urmează denumirii fișierului executabil . In cazul execuției programului de sub IDE , ei vor fi scriși în Run|Parameters .

Numărul de parametrii din linia de comandă este stabilit cu funcția PARAMCOUNT . Acești parametrii vor fi citiți ca niște date de tip STRING , prin intermediul funcției PARAMSTR(i) , unde i este numărul de ordine al parametrului (i=1,2,...) . In cazul în care i=0 , funcția PARAMSTR va returna numele fișierului executat , împreună cu adresa lui pe disc .

Programul cântă diferite note muzicale , care îi sunt date în linia de comandă . Acest lucru este realizat de către difuzorul calculatorului , prin intermediul comenzilor SOUND (pornește

difuzorul) , DELAY (introduce o pauză) şi NOSOUND (opreşte difuzorul) , comenzi ale unit-ului crt .

Denumirile notelor şi frecvenţele sunetelor respective sunt stocate în două matrici , definite în secţiunea CONST .

Notaţia string[n] defineşte un tip de dată STRING , cu un număr maxim de n caractere . Tipul standard STRING are o lungime de 255 de caractere .

Iată programul :

```
PROGRAM canta;
uses crt;
const nota:array[1..7] of string[3]=
        ('do','re','mi','fa','sol','la','si');
      frecventa:array[1..7] of word=
                (261,293,329,349,392,440,494);
var i,j,k:integer; n:string[3]; f:word;
begin
     k:=paramcount;
     {nr de parametrii dati in linia de comanda}
     for i:=1 to k do
     {pentru toate notele din linia de comanda}
     begin
          n:=paramstr(i);
          {nota muzicala de executat}
          for j:=1 to 7 do
          if n=nota[j]
          then begin
               f:=frecventa[j];
               sound(f);
               delay(500);
               nosound;
               end;
     end;
end.
```

Iată un exemplu de utilizare a acestui program : tastaţi din linia de comandă :

canta do re mi<ENTER>

<u>crtp.pas</u>

Programul următor vă arată un alt mod , mai rafinat , de a citi date de la tastatură . Funcţia READKEY face o pauză până când este apăsată o tastă , fiind citit un caracter . Funcţia KEYPRESSED interoghează tastatura : A fost o tastă apăsată ? Dacă da , răspunsul este TRUE , dacă nu , răspunsul este FALSE .

Astfel , în locul instrucţiunii de aşteptare

READLN;

putem folosi comanda

REPEAT UNTIL KEYPRESSED;

sau comanda mai restrictivă

REPEAT UNTIL READKEY=c;

unde c este o constantă sau o variabilă de tipul CHAR .

In funcție de tasta apăsată programul afișează un caracter , trece la o altă linie pentru afișarea caracterelor sau își încheie execuția . Această selecție este făcută cu ajutorul comenzii CASE . Scrierea caracterelor se face folosind codul lor ASCII .

Ieșirea din program , mai devreme decât instrucțiunea finală END. , se poate face cu comanda HALT . O altă comandă , mai flexibilă , este EXIT . In cazul în care ea apare într-o procedură sau o funcție , ea produce ieșirea din ele , programul reîntorcându-se în locul de unde a fost apelată respectiva procedură sau funcție . In cazul în care instrucțiunea EXIT apare în programul principal , ea cauzează terminarea execuției acestuia , ca și instrucțiunea HALT .

Repetarea testării tastaturii se face cu un salt GOTO la o etichetă ce a fost definită în secțiunea LABEL .

Iată programul :

```
PROGRAM crtp;
uses crt;
label repeta;
var c:char;
begin
  writeln('Acest program deseneaza caracterele');
  writeln('cupa , romb , trefla , pica');
  writeln('pina cind se apasa tasta <q>');
  writeln('Apasati tastele :');
  writeln('<c> cupa , <r> romb , ',
        '<t> trefla , <p> pica');
  writeln('Celelalte taste trec la alta linie');
  writeln;
repeta:
     c:=readkey;
     case c of
     'c': write(#3);
     'r': write(#4);
     't': write(#5);
     'p': write(#6);
     'q': halt;
     else writeln   {trece pe alta linie}
     end;
     goto repeta;
end.
```

Scrieți un program care să scrie pe ecran următoarele date de tipul CHAR : 'a' și #97 . Simbolul # indică faptul că respectivul caracter este dat prin intermediul codului său ASCII (Americam Standard Code for Information Interchange) . Cele două date de tipul CHAR sunt echivalente .

Scrieți un program în care instrucțiunea de

așteptare READLN; să fie înlocuită de instrucțiunea REPEAT UNTIL READKEY=#13; . In ambele cazuri apăsarea tastei <ENTER> provoacă terminarea pauzei , 13 fiind codul ASCII pentru <ENTER> .

<u>fisier1.pas</u>

Până acum programele au primit date de la tastatură , și au afișat date pe monitor . Insă mai există un mod de introducere-scoatere de date , și anume prin intermediul fișierelor . Acest mod este util când datele sunt foarte numeroase , când trebuie transportate de la un calculator la altul , când prelucrările numerice trebuiesc repetate de mai multe ori asupra lor .

Pentu acest tip de operații limbajul Pascal prevede tipurile de date FILE , FILE OF ... și TEXT .

Fișierele de tip TEXT conțin linii de text . Citirea din ele se face cu comanda READLN , iar scrierea în ele se face cu comanda WRITELN .

Din fișierele de tip FILE și FILE OF ... datele se citesc cu comanda READ , și se scriu cu comanda WRITE .

De fiecare dată când citim sau scriem într-un fișier trebuie ca primul parametru al instrucțiunii să fie variabila de tip fișier din care se scrie sau se citește . Altfel citirea se face de la tastatură , iar scrierea se face pe monitor . Tastatura este echivalentă cu fișierul INPUT , de tip TEXT , iar monitorul este echivalent cu fișierul OUTPUT , de tip TEXT . Folosind unit-ul PRINTER se poate folosi fișierul LST , de tip TEXT , echivalent cu imprimanta , pentru scrierea la imprimantă .

Variabila de tip fișier este pusă în corespondență cu un nume de fișier prin intermediul comenzii ASSIGN .

Comanda REWRITE creează un fișier nou , și îl deschide , pregătindu-l pentru scriere . Din acest fișier vor putea fi citite date numai după ce ele au fost în prealabil scrise .

Comanda RESET deschide un fișier deja existent pe disc, pregătindu-l pentru citire sau scriere .

Fiecare fișier proaspăt deschis are un cursor poziționat la începutul lui . Pe măsură ce datele sunt citite sau scrise , acest cursor înaintează spre capătul fișierului . Datele sunt citite sau scrise la poziția cursorului din fișier . Când toate datele au fost citite , cursorul fiind la capătul fișierului , orice încercare de a mai citi date din fișier va fi sortită eșecului . Pentru a evita această eroare programul Pascal ne pune la dispoziție funcția EOF (End Of File) ce ne indică dacă avem sau nu cursorul poziționat la sfârșitul fișierului specificat .

Pentru fișierele cu date de un anume tip (FILE OF ...) există următoarele comenzi : SEEK (poziționează cursorul în fișier) , FILEPOS (returnează poziția cursorului în fișier) , FILESIZE (returnează numărul de elemente ale fișierului) .

Comanda CLOSE închide fișierul , toate modificările făute asupra acestuia fiind salvate pe disc .

Programul următor creează un fișier în care stochează 5 caractere .

```
PROGRAM fisier1;
var nume:string;
    f:file of char;
    i:integer;
    c:char;
begin
    writeln('Acest program creeaza un fisier');
    writeln('in care stocheaza 5 caractere');
    writeln('Introduceti numele fisierului');
    write('nume = ');
    readln(nume);
    assign(f,nume);
    rewrite(f);
    writeln('Introduceti caracterele , pe rind');
    for i:=1 to 5 do
    begin
         write('caracterul nr ',i,' este    ');
         readln(c);
         write(f,c);
    end;
    close(f);
end.
```

fisier2.pas

Programul următor citește din fișierul creat de programul precedent cele 5 caractere .

De obicei utilizatorul nu știe câte date sunt scrise într-un fișier . S-ar putea chiar ca fișierul să nu conțină nici o dată . In cazul deschiderii unui fișier ce nu are nici o dată cursorul este poziționat la începutul lui , dar această poziție coincide și cu sfârșitul lui . De aceea e indicat ca înainte de a se efectua citirea dintr-un astfel de fișier să se verifice dacă există într-adevăr o dată de citit la poziția cursorului , dacă nu cumva cursorul fișierului a ajuns la sfârșitul acestuia . Această verificare o realizăm cu funcția EOF .

Pentru a afla și numărul de date scrise în fișier se inițializează variabila i cu 0 , și după fiecare citire din fișier o mărim cu o unitate .

Cea mai performantă structură ce repetă citirea unui element din fișier , după ce în

prealabil s-a asigurat că nu avem cursorul fișierului poziționat la sfârșitul lui , este formată cu ajutorul comenzii WHILE ... DO ... , și este folosită de programul prezentat .

Iată programul :

```
PROGRAM fisier2;
var nume:string;
    f:file of char;
    i:integer;
    c:char;
begin
     writeln('Acest program deschide un fisier');
     writeln('din care citeste caractere');
     writeln('Introduceti numele fisierului');
     write('nume = ');
     readln(nume);
     assign(f,nume);
     reset(f);
     i:=0;
     while not(eof(f)) do
     begin
          i:=i+1;
          write('caracterul nr ',i,' este   ');
          read(f,c);
          writeln(c);
     end;
     close(f);
     writeln('Nr de caractere din fisier e ',i);
     readln;
end.
```

fisier3.pas

Acest program este foarte asemanător cu cel precedent , numai că structura WHILE ... DO ... a fost înlocuită cu structura REPEAT ... UNTIL ... . Programul iese din ciclul astfel creat atunci când au fost citite toate datele din fișier . Totuși , deoarece condiția EOF este testată după ce a fost realizată prima citire din fișier , există riscul ca tocmai această primă citire să producă o eroare , atunci când fișierul nu conține nici o dată . De aceea este de preferat varianta prezentată în programul precedent , programul prezent putând fi folosit doar atunci când avem certitudinea că există cel puțin o dată de citit în fișier .

Iată programul :

```
PROGRAM fisier3;
var nume:string;
    f:file of char;
    i:integer;
    c:char;
begin
     writeln('Acest program deschide un fisier');
```

```
     writeln('din care citeste caractere');
{programul functioneaza incorect in cazul in care
 fisierul nu are nici un caracter scris in el}
     writeln('Introduceti numele fisierului');
     write('nume = ');
     readln(nume);
     assign(f,nume);
     reset(f);
     i:=0;
     repeat
           i:=i+1;
           write('caracterul nr ',i,' este   ');
           read(f,c);
           writeln(c);
     until eof(f);
     close(f);
     writeln('Nr de caractere din fisier e ',i);
     readln;
end.
```

demoa.pas , demob.pas

In continuare exemplificăm scrierea unui unit de către utilizator . Unit-ul demob.pas are o singură funcție , și anume ridicarea la putere , cu exponent real , pe care am utilizat-o și în demo4.pas . Programul demoa.pas este foarte asemănător cu programul demo4.pas , doar că nu mai conține definiția funcției , pe care o ia din unit .

Iată cele două programe :

```
PROGRAM demoA;
uses demoB;
var x,y:real;

begin
     writeln('Programul calculeaza pe x^y');
     writeln('x,y sunt numere reale si x>0');
     write('x = ');
     readln(x);
     write('y = ');
     readln(y);
     writeln('x^y = ',putere(x,y));
     readln;
end.


UNIT demoB;

INTERFACE

FUNCTION putere(baza,exponent:real):real;

IMPLEMENTATION

FUNCTION putere;
```

```
begin
    putere:=exp(exponent*ln(baza));
end;

END.
```

Schimbați destinația programului compilat , prin opțiunea Compile|Destination|Disk , astfel ca la compilarea programului demob.pas , pe disc să apară fișierul demob.tpu . Abia după ce a fost creat acest fișier , se poate compila și programul principal demoa.pas , rezultând pe disc fișierul executabil demoa.exe .

CAP II. GRAFICA IN TURBO PASCAL

1) Modul alfanumeric si modul grafic

Calculatoarele pot folosi ecranul în două feluri : în modul alfanumeric , sau în modul grafic . In modul alfanumeric ecranul este împărțit în linii și coloane , în intersecția unei linii cu o coloană putându-se tipări un caracter (literă , cifră , simbol matematic , tot ce intră sub tipul CHAR) . In modul grafic împărțirea este mai fină, fiecare punct (pixel) de pe ecran fiind caracterizat de o coordonată orizontală și una verticală .

Atât în primul caz , cât și în cel de-al doilea , coordonata orizontală crește de la stânga la dreapta , iar coordonata verticală crește de sus în jos . In modul alfanumeric coordonata colțului din stânga-sus este (1,1) , iar în modul grafic ea este (0,0) .

In cazul în care este definită pe ecran o fereastră , paragraful de mai sus se referă la această ferestră , și nu la întreg ecranul .

2) Modul alfanumeric (modul text)

Acesta este modul în care calculatorul lucrează în mod obișnuit . Introducerea datelor de la tastatură se face prin comenzile READ și READLN , iar afișarea datelor pe ecran se face prin comenzile WRITE și WRITELN , comenzi ale unitului SYSTEM . Poziționarea cursorului se face cu comanda GOTOXY , iar aflarea coordonatelor acestuia se face cu comenzile WHEREX și WHEREY , comenzi ale unitului CRT . Stabilirea culorii textului se face cu comanda TEXTCOLOR , iar cea a fondului cu comanda TEXTBACKGROUND . Comanda WINDOW definește o fereastră text pe ecran .

Modul alfanumeric este caracterizat de un specificator , numit mod , care depinde de numărul maxim de linii și de coloane , precum și de posibilitățile cromatice (alb-negru sau color) . Comanda TEXTMODE selectează modul text specificat .

3) Modul grafic

Modul grafic este inițializat cu comanda INITGRAPH , și este abandonat cu comanda CLOSEGRAPH , comenzi ale unitului GRAPH . Prin aceste comenzi se încarcă , respectiv se scoate din memorie , un program (driver grafic) care se ocupă de interfața program Pascal - structura hard a calculatorului . Driverul grafic depinde deci de calculator . In cazul în care dorim să

facem treceri între modul alfanumeric și modul grafic , pentru a nu încărca și scoate mereu din memorie driverul grafic , vom folosi comenzile RESTORECRTMODE și SETGRAPHMODE , care lasă driverul grafic prezent în memorie .

Introducerea datelor de la tastatură se face cu comanda READKEY , comandă a unitului CRT , iar afișarea datelor pe ecran se face cu comenzile OUTTEXT și OUTTEXTXY , comenzi ale unitului GRAPH . Poziționarea cursorului se face cu comanda MOVETO , iar aflarea coordonatelor acestuia se face cu comenzile GETX și GETY , comenzi ale unitului GRAPH . Stabilirea culorii de desenare se face cu comanda SETCOLOR , iar cea a fondului cu comanda SETBKCOLOR . Comanda SETVIEWPORT definește o fereastră pe ecran .

Ca și în modul alfanumeric , și în modul grafic vom avea un specicator numit mod , ce este selectat odată cu inițializarea modului grafic .

4) <u>Comenzi importante ale unitului GRAPH</u>

INITGRAPH(GraphDriver:integer; GraphMode:integer; PathToDriver:string)

Inițializează modul grafic , încărcând în memorie driverul grafic .

Pentru ca programul să determine singur driverul și modul , vom face inițializarea :

driver:=detect;

PathToDriver este calea de acces la driverul grafic (la fișierele *.bgi) .

GRAPHRESULT:integer

Este o functie ce returnează codul de eroare al ultimei operații grafice .

GRAPHERRORMSG(ErrorCode:integer):string

Returnează mesajul de eroare corespunzător codului de eroare dat de funcția precedentă .

GETMAXX:integer

Este o funcție ce returnează coordonata orizontală maximă pe care o poate avea un pixel . (639 pentru VGA)

GETMAXY:integer

Este o funcție ce returnează coordonata verticală maximă pe care o poate avea un pixel . (479 pentru VGA)

RECTANGLE(x1,y1,x2,y2:integer)

Desenează un dreptunghi , definit de colțurile diagonale (x1,y1) și (x2,y2) , date în pixeli , folosind culoarea și stilul de linie curente .

LINE(x1,y1,x2,y2:integer)

Desenează un segment de dreaptă , între

punctele (x1,y1) și (x2,y2) , date în pixeli .

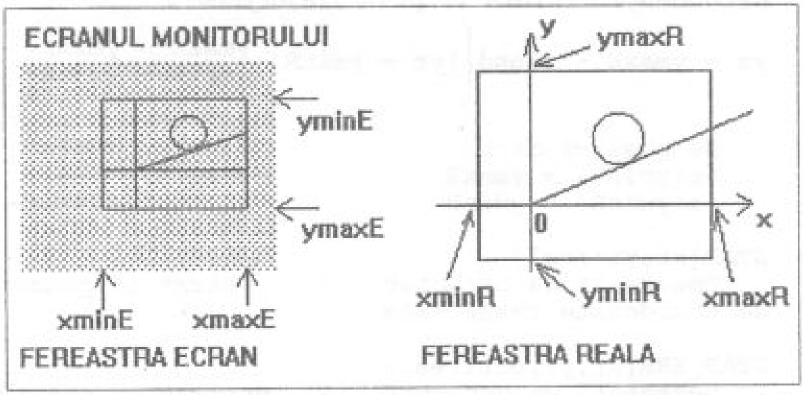
PUTPIXEL(x,y:integer; color:word)
Desenează un punct , de culoare color , la (x,y) .

OUTTEXTXY(x,y:integer; TextString:string)
Desenează șirul de caractere TextString pe ecran , astfel încât colțul din stânga sus al primului caracter să se afle la (x,y) .

CLOSEGRAPH
Inchide modul grafic și restabilește modul alfanumeric .

5) Programul PASCAL

In continuare este prezentat unit-ul GRAPHICS, ce va fi folosit în programele viitoare , ce are ca principală funcție reprezentarea grafică pe ecran , într-o fereastră dreptunghiulară de colțuri (XminE , YminE) și (XmaxE , YmaxE) date în pixeli , a unor puncte de coordonate reale , ce aparțin unei ferestre dreptunghiulare de colțuri (XminR , YminR) și (XmaxR , YmaxR) date în numere reale .



Coordonata orizontală (în pixeli) pe ecran a punctelor crește de la stânga la dreapta , la fel ca și în reprezenterea grafică a planului real .
Coordonata verticală (în pixeli) pe ecran a punctelor crește de sus în jos , invers decât în reprezentarea grafică a planului real .
Datorită acestei diferențe , formulele de trecere de la coordonatele reale la cele în pixeli vor fi diferite pentru cele două axe (Ox și Oy) .
Iată procedurile și funcțiile acestui unit :

INIT_GRAPH_MODE
Inițializează modul grafic , determină

coordonata maximă pe axa x (maxX) şi pe axa y (maxY) .

INIT_WINDOW(x1e,x2e,y1e,y2e:integer;
            x1r,x2r,y1r,y2r:real)

Iniţializează valorile ce definesc fereastra ecran şi fereastra reală , încadrează fereastra ecran într-un dreptunghi , trasează axele de coordonate (Ox şi Oy) dacă acestea apar în fereastra reală .

XE(xr:real):integer

Face corespondenţa între coordonata orizontală reală şi coordonata orizontală ecran , printr-o dependenţă liniară şi prin rotunjire :

$$xe = xminE + round\left[(xr - xminR)\cdot\frac{xmaxE - xminE}{xmaxR - xminR}\right] \quad (1)$$

Se observă că :
xe(xminR) = xminE
xe(xmaxR) = xmaxE

YE(yr:real):integer

Face corespondenţa între coordonata verticală reală şi coordonata verticală ecran , printr-o dependenţă liniară şi prin rotunjire :

$$ye = ymaxE - round\left[(yr - yminR)\cdot\frac{ymaxE - yminE}{ymaxR - yminR}\right] \quad (2)$$

Se observă că :
ye(yminR) = ymaxE
ye(ymaxR) = yminE

STAR(xr,yr:real)

Desenează un caracter '*' centrat în punctul de coordonate reale date .

STAR_ERR(xr,yr,err:real)

Desenează un caracter '*' centrat în punctul de coordonate reale date , trasând şi bara corespunzătoare erorii standard ±err .

PUNCT_HISTO(xr,yr:real)

Desenează un segment de dreaptă orizontal de lungime reală histo , centrat în punctul de coordonate reale date . Inainte de utilizare trebuie iniţializată valoarea variabilei histo .

GRAPH_FUNCTIE(f:func; color:integer)

Desenează graficul funcţiei reale , de argument real :
y = f(x)
calculând şi desenând , pentru fiecare pixel de pe orizontală , un număr de puncte egal cu

YperX . Variabila YperX este inițializată cu valoarea 1 , dar pentru funcții cu o variație rapidă , pentru ca graficul să dea senzația de continuitate , se vor folosi valori mai mari pentru YperX .

Deoarece această procedură are ca parametru de intrare o funcție , programul principal ce o folosește trebuie să fie compilat cu opțiunea {$F+} (Force Far Calls) . Funcția al cărei grafic dorim să-l trasăm trebuie să fie de tipul func , sau altfel spus cu un argument real și cu o valoare reală .

Iată în continuare programul Pascal pentru unit-ul graphics , și cel pentru un program exemplificativ ce folosește acest unit :

```
UNIT GRAPHICS;

interface

uses graph;
const YperX:integer=1;
type func=function(x:real):real;
var XminE,XmaxE,YminE,YmaxE,maxX,maxY:integer;
    XminR,XmaxR,YminR,YmaxR,scaraX,scaraY:real;
    histo:real;
procedure init_graph_mode;
procedure init_window(x1e,x2e,y1e,y2e:integer;
                      x1r,x2r,y1r,y2r:real);
function xe(xr:real):integer;
function ye(yr:real):integer;
procedure star(xr,yr:real);
procedure star_err(xr,yr,err:real);
procedure punct_histo(xr,yr:real);
procedure graph_functie(f:func;color:integer);

implementation

PROCEDURE INIT_GRAPH_MODE;
var driver,mode,errorcode:integer;
begin
driver:=detect; initgraph(driver,mode,'');
errorcode:=graphresult;
if errorcode<>grok then begin
  writeln('Eroare de grafica: ',
          grapherrormsg(errorcode));
  halt; end;
maxX:=getmaxx;{de regula 639}
maxY:=getmaxy;{de regula 479}
end;

PROCEDURE INIT_WINDOW;
begin
if (x1e<0) or (x2e>maxX) or (y1e<0) or (y2e>maxY)
then halt;
```

```
XminE:=x1e; XmaxE:=x2e; YminE:=y1e; YmaxE:=y2e;
XminR:=x1r; XmaxR:=x2r; YminR:=y1r; YmaxR:=y2r;
scaraX:=(XmaxE-XminE)/(XmaxR-XminR);
scaraY:=(YmaxE-YminE)/(YmaxR-YminR);
rectangle(XminE,YminE,Xmaxe,YmaxE);
if (0>XminR) and (0<XmaxR)
then line(xe(0),YminE,xe(0),YmaxE);
if (0>YminR) and (0<YmaxR)
then line(XminE,ye(0),XmaxE,ye(0));
end;

FUNCTION XE;
begin
if xr<XminR then begin xe:=XminE; exit; end;
if xr>XmaxR then begin xe:=XmaxE; exit; end;
xe:=XminE+round((xr-XminR)*scaraX);
end;

FUNCTION YE;
begin
if yr<YminR then begin ye:=YmaxE; exit; end;
if yr>YmaxR then begin ye:=YminE; exit; end;
ye:=YmaxE-round((yr-YminR)*scaraY);
end;

PROCEDURE STAR;
begin outtextxy(xe(xr)-3,ye(yr)-3,'*'); end;

PROCEDURE STAR_ERR;
var x0,y0,yj,ys:integer;
begin
x0:=xe(xr); y0:=ye(yr);
ys:=ye(yr-err); yj:=ye(yr+err);
outtextxy(x0-3,y0-3,'*'); line(x0,yj,x0,ys);
line(x0-3,yj,x0+3,yj); line(x0-3,ys,x0+3,ys);
end;

PROCEDURE PUNCT_HISTO;
var y0,xs,xd:integer;
begin
y0:=ye(yr);
xs:=xe(xr-histo/2); xd:=xe(xr+histo/2);
line(xs,y0,xd,y0);
end;

PROCEDURE GRAPH_FUNCTIE;
var xr,dx:real; i:integer;
begin
dx:=1/(YperX*scaraX); xr:=XminR;
repeat
  putpixel(xe(xr),ye(f(xr)),color);
  xr:=xr+dx;
until xr>XmaxR;
end;

end.
```

```
{$F+}
PROGRAM USEGRAPH;
uses graph,graphics,crt;

FUNCTION F(x:real):real;
begin f:=sin(x); end;

begin
init_graph_mode;
init_window(400,600,100,200,-10,10,-2,2);
YperX:=5; graph_functie(f,red);
repeat until readkey='q';
closegraph;
end.
```

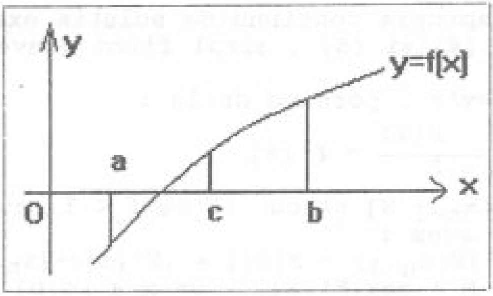
6) Problema

Studiați cu atenție , cu ajutorul Help-ului din Turbo Pascal , toate procedurile și funcțiile ce au apărut în acest capitol .

CAP III. REZOLVAREA ECUATIILOR

1) Metoda injumatatirii intervalului

Metoda înjumătăţirii intervalului este cea mai simplă metodă de rezolvare a unei ecuaţii . Ea se aplică atunci când cunoaştem faptul că ecuaţia , scrisă sub forma $f(x) = 0$, are o rădăcină , şi numai una , într-un interval [a,b] .



Graficul funcţiei f(x) înţeapă axa Ox în rădăcina căutată , punctele din capetele intervalului aflându-se unul mai sus şi celălalt mai jos de axa Ox . Vom avea :

$$f(a)\cdot f(b) \le 0 \quad (1)$$

Metoda presupune micşorarea intervalului [a,b] , având grijă ca el să conţină în continuare rădăcina ecuaţiei , până când lungimea lui , care este o măsură a preciziei cu care soluţia a fost găsită , este mai mică decât o valoare dată . Micşorarea intervalului [a,b] se face prin alegerea punctului $c = \frac{a+b}{2}$ din mijlocul lui , prin testarea subintervalului în care se află rădăcina , şi prin definirea noului interval ca fiind jumătate din cel precedent . Astfel :

$f(a)\cdot f(c) \le 0 \Rightarrow$ rădăcina căutată este în [a,c]
$\Rightarrow$ [a,b] trece în [a,c] (b:=c) (2)
$f(c)\cdot f(b) \le 0 \Rightarrow$ rădăcina căutată este în [c,b]
$\Rightarrow$ [a,b] trece în [c,b] (a:=c) (3)

Deoarece cazurile (2) şi (3) se exclud unul pe celălalt , funcţia având numai o rădăcină în intervalul [a,b] , programul va testa numai o inegalitate , introducând o ramificaţie IF .. THEN .. ELSE .

Injumătăţirea intervalului se opreşte atunci când lungimea lui [a,b] a devenit mai mică decât precizia impusă soluţiei , rădăcina căutată fiind aproximată de $\frac{a+b}{2}$.

2) Metoda iteratiilor

Fie funcția $f(x)$, care are o singură rădăcină în intervalul $[a,b]$. Metoda iterațiilor presupune existența unei funcții $F(x)$, cu proprietatea că $|F'(x)| < 1$ în intervalul $[a,b]$, astfel încât ecuațiile (4) și (5) să fie echivalente .

$$f(x) = 0 \qquad (4)$$

$$F(x) = x \qquad (5)$$

In aceste condiții , plecând de la o aproximație inițială x_1 , și generând șirul :

$$x_{n+1} = F(x_n)$$

ne vom apropia continuu de soluția exactă X a ecuațiilor (4) și (5) , șirul fiind convergent către ea .

Intr-adevăr , pornind de la :

$$\frac{F(x_{n-1}) - F(X)}{x_{n-1} - X} = F'(z) \qquad (6)$$

cu $z \in [x_{n-1},X]$ și cu $|F'(z)| < 1$ comform ipotezei , avem : (7)

$$|x_n - X| = |F(x_{n-1}) - F(X)| = |F'(z)|\cdot|x_{n-1} - X|$$

Notând $M = \max|F'(x)|$, cu $x \in [a,b]$, $M < 1$, din (7) vom avea :

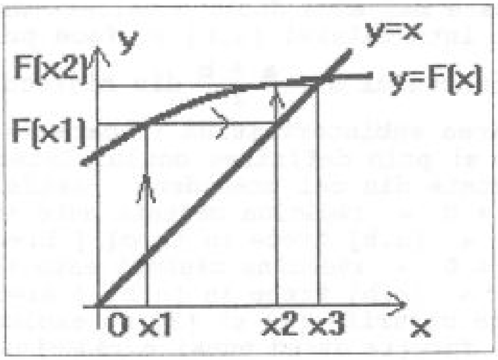
$$|x_n - X| \le M\cdot|x_{n-1} - X| \qquad (8)$$

Prin inducție obținem :

$$|x_n - X| \le M^{n-1}\cdot|x_1 - X| \qquad (9)$$

Deoarece $M < 1$, se vede clar că în limita $n\to\infty$ vom avea $x_n \to X$.

O ilustrare geometrică a condiției $|F'(x)| < 1$ pentru ca procesul de iterație să fie convergent este dată în figura următoare :



Iterația se oprește când diferența $|x_n - x_{n-1}|$ este mai mică decât precizia cu care căutăm soluția aproximativă . Este bine să se prevadă și o ieșire forțată din ciclul iterațiilor dacă numărul de iterații depășește o anumită valoare maximă .

Această metodă poate fi generalizată și la cazul a două ecuații cu două necunoscute . Se caută două funcții F1(x,y) și F2(x,y) astfel

încât cele două sisteme să fie echivalente :

$$\begin{cases} f1(x,y) = 0 \\ f2(x,y) = 0 \end{cases} \quad (10)$$

$$\begin{cases} x = F1(x,y) \\ y = F2(x,y) \end{cases} \quad (11)$$

In cazul în care , pentru punctele unei vecinătăţi a soluţiei căutate , avem îndeplinite condiţiile :

$$\left|\frac{\partial F1(x,y)}{\partial x}\right| + \left|\frac{\partial F1(x,y)}{\partial y}\right| < 1 \quad (12)$$

$$\left|\frac{\partial F2(x,y)}{\partial x}\right| + \left|\frac{\partial F2(x,y)}{\partial y}\right| < 1$$

pornind de la o soluţie aproximativă (x_1,y_1) , şirul generat de :

$$x_{n+1} = F1(x_n,y_n) \quad (13)$$
$$y_{n+1} = F2(x_n,y_n)$$

converge către soluţia exactă .

3) Metoda tangentei (metoda Newton-Raphson)

Fie o ecuaţie $f(x) = 0$.

Pentru aplicarea acestei metode este necesar ca în vecinătatea soluţiei să fie satisfăcute următoarele condiţii :

a) să existe o singură soluţie

b) prima derivată a funcţiei să nu se anuleze

c) a doua derivată să existe şi să fie mărginită

Observăm că aceste condiţii sunt mai puţin restrictive decât cele de la metoda iteraţiilor , unde trebuia să avem $|F'(x)| < 1$.

Fie X soluţia exactă , şi x_n o aproximaţie iniţială . Dezvoltând în serie Taylor pe f(X) în jurul lui $f(x_n)$ şi păstrând doar primii doi termeni vom obţine :

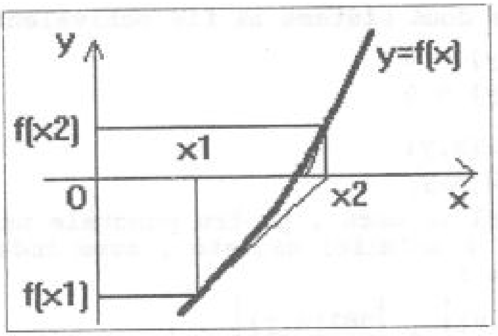
$$0 = f(X) \approx f(x_n) + (X-x_n)\cdot f'(x_n) \quad (14)$$

De aici îl vom scoate pe X , care însă este doar o aproximaţie mai bună x_{n+1} , şi nu soluţia exactă , datorită truncherii seriei Taylor .

$$0 = f(x_n) + (x_{n+1}-x_n)\cdot f'(x_n) \quad (14')$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (15)$$

Rezolvarea unei ecuaţii prin această metodă poate fi uşor înţeleasă din următoarea figură :



Pentru o valoare aproximativă x1 a soluției , se intersectează tangenta în punctul $(x1, f(x1))$ la curba $y = f(x)$ cu axa Ox , obținându-se x2 . Intr-adevăr , calculând tangenta în x1 obținem :

$$f'(x1) = \frac{f(x2) - f(x1)}{x2 - x1} = \frac{-f(x1)}{x2 - x1} \Rightarrow$$

$$\Rightarrow \quad f'(x_n) = \frac{-f(x_n)}{x_{n+1} - x_n} \Rightarrow \quad (15)$$

Metoda se caracterizează printr-o convergență foarte rapidă . Procesul iterativ se repetă până când se obține precizia dorită , adică $|x_n - x_{n-1}|$ mai mic decât precizia .

Pentru a nu fi necesar în (15) să se evalueze și expresia analitică a lui $f'(x)$, formula de recurență poate fi scrisă sub forma :

$$x_{n+1} = x_n - \frac{f(x_n)\cdot(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \qquad (16)$$

în care derivata a fost calculată numeric :

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \qquad (17)$$

4) Programele PASCAL

In continuare sunt date trei programe , ce rezolvă ecuația :

$$x^2 = \frac{1}{2}\cdot\sin(x)$$

prin cele trei metode prezentate .

Intervalul în care se caută rădăcina este $(0, \frac{\pi}{6})$. Pentru metoda iterațiilor funcția este :

$$F(x) = \frac{\sin(x)}{2\cdot x}$$

Aproximația inițială pentru metoda iterațiilor și metoda tangentei a fost aleasă 0.3 .

```
PROGRAM injumatatire;
var a,b,c,eps:real;
    it,itmax:integer;

FUNCTION f(x:real):real;
begin
     f:=x*x-1/2*sin(x);
end;

begin
     eps:=1E-5;
     itmax:=1000;
     it:=0;
     a:=0.00001;
     b:=0.6;
     repeat
           c:=(a+b)/2;
           if f(a)*f(c)<=0
             then b:=c
             else a:=c;
           it:=it+1;
     until (b-a<eps) or (it>itmax);
     writeln(c);
     readln;
end.


PROGRAM iteratii;
var xn,xv,eps:real;
    it,itmax:integer;

FUNCTION F(x:real):real;
begin
     F:=sin(x)/(2*x);
end;

begin
     eps:=1E-5;
     itmax:=1000;
     it:=0;
     xn:=0.3;
     repeat
           xv:=xn;
           xn:=F(xv);
           it:=it+1;
     until (abs(xn-xv)<eps) or (it>itmax);
     writeln(xn);
     readln;
end.


PROGRAM tangenta;
var xn,xv,eps:real;
    it,itmax:integer;

FUNCTION f(x:real):real;
begin
```

```
     f:=x*x-1/2*sin(x);
end;

FUNCTION df(x:real):real;
begin
     df:=2*x-1/2*cos(x);
end;

begin
     eps:=1E-5;
     itmax:=1000;
     it:=0;
     xn:=0.3;
     repeat
           xv:=xn;
           xn:=xv-f(xv)/df(xv);
           it:=it+1;
     until (abs(xn-xv)<eps) or (it>itmax);
     writeln(xn);
     readln;
end.
```

5) Probleme

1) Modificați ultimul program , astfel încât derivata să fie calculată numeric .

2) Să se scrie un program de calcul al radicalului de ordin 5 din A folosind o metodă iterativă și fără a folosi alți operatori decît cei de adunare, înmulțire , scădere , împărțire .

CAP IV. DERIVAREA SI INTEGRAREA NUMERICA

1) Dezvoltarea in serie Taylor

O funcție continuă și infinit derivabilă se dezvoltă astfel :

$$f(y) = f(x) + f'(x)\cdot(y-x) + \frac{1}{2}\cdot f''(x)\cdot(y-x)^2 + ..$$

$$.. + \frac{1}{n!}\cdot f^{(n)}(x)\cdot(y-x)^n + .. \qquad (1)$$

unde $f^{(n)}(x)$ este derivata de ordinul n a funcției f în punctul x . Pentru valori mici ale diferenței (y-x) ne putem opri la primii termeni ai seriei , cu o bună aproximație .

2) Derivarea numerica

Derivarea numerică este indicată în cazul în care funcția f(x) este dată printr-un tabel de valori $f(x_i)$ corespunzător unei diviziuni a intervalului [a,b] în care este cunoscută funcția , sau dacă expresia analitică a derivatei este complicată .

Oprindu-ne la primii doi termeni ai seriei vom avea :

$$f(x+h) \approx f(x) + f'(x)\cdot h$$

De aici obținem :

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \qquad (2)$$

Această formulă presupune cunoașterea valorilor funcției în puncte mai mari decât cel în care calculăm derivata .

Analog :

$$f(x-h) \approx f(x) - f'(x)\cdot h$$

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \qquad (2')$$

Această formulă presupune cunoașterea valorilor funcției în puncte mai mici decât cel în care calculăm derivata .

In cazul în care cunoaștem valorile funcției în intervalul [a,b] , pentru a calcula derivata în punctul a se folosește formula (2) , iar pentru a calcula derivata în punctul b se folosește formula (2') . Pentru punctele intermediare se poate folosi una din cele două formule , ambele având aceeași precizie .

In limita $h\to0$ din (2) obținem chiar definiția derivatei .

O precizie mai bună putem obține dacă ne oprim la primii trei termeni ai seriei .

$$f(x+h) \approx f(x) + f'(x)\cdot h + \frac{1}{2}\cdot f''(x)\cdot h^2$$

$$f(x-h) \approx f(x) - f'(x)\cdot h + \frac{1}{2}\cdot f''(x)\cdot h^2$$

Privim aceste ecuații ca un sistem cu două necunoscute , $f'(x)$ și $f''(x)$. Obținem :

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2\cdot h} \qquad (3)$$

$$f''(x) \approx \frac{f(x+h) + f(x-h) - 2\cdot f(x)}{h^2} \qquad (4)$$

Pentru a putea calcula derivatele și în capetele a și b ale intervalului [a,b] în care cunoaștem funcția , vom alege alte ecuații .

Dacă știm valori ale funcției în puncte mai mari decât cel în care calculăm derivata :

$$f(x+h) \approx f(x) + f'(x)\cdot h + \frac{1}{2}\cdot f''(x)\cdot h^2$$

$$f(x+2\cdot h) \approx f(x) + f'(x)\cdot(2\cdot h) + \frac{1}{2}\cdot f''(x)\cdot(2\cdot h)^2$$

Din rezolvarea sistemului rezultă :

$$f'(x) \approx \frac{4\cdot f(x+h) - 3\cdot f(x) - f(x+2\cdot h)}{2\cdot h} \qquad (3')$$

$$f''(x) \approx \frac{f(x) + f(x+2\cdot h) - 2\cdot f(x+h)}{h^2} \qquad (4')$$

Dacă știm valori ale funcției în puncte mai mici decât cel în care calculăm derivata :

$$f(x-h) \approx f(x) - f'(x)\cdot h + \frac{1}{2}\cdot f''(x)\cdot h^2$$

$$f(x-2\cdot h) \approx f(x) - f'(x)\cdot(2\cdot h) + \frac{1}{2}\cdot f''(x)\cdot(2\cdot h)^2$$

Din rezolvarea sistemului rezultă :

$$f'(x) \approx \frac{3\cdot f(x) + f(x-2\cdot h) - 4\cdot f(x-h)}{2\cdot h} \qquad (3'')$$

$$f''(x) \approx \frac{f(x) + f(x-2\cdot h) - 2\cdot f(x-h)}{h^2} \qquad (4'')$$

Toate aproximațiile de mai sus au fost făcute în ipoteza h mic . Erorile introduse prin aceste aproximații provin din termenii neglijați din dezvoltarea în serie Taylor .

3) Integrarea numerica

Fie o funcție continuă $f(x)$, și o integrală definită de forma :

$$I = \int_a^b f(x)\cdot dx$$

Această integrală poate fi calculată numeric prin mai multe metode , toate apelând la

semnificația integralei definite de a fi mărimea suprafeței limitată de $x = a$, $x = b$, $y = f(x)$ și $y = 0$.

Se împarte segmentul [a,b] în N intervale egale , prin punctele echidistante x_0 , x_1 , x_2 , .. , x_N , având $x_0 = a$ și $x_N = b$, iar suprafața I se consideră egală cu suma suprafețelor elementare formate , limitate de $x = x_i$, $x = x_{i+1}$, $y = f(x)$ și $y = 0$.

$$I = \int_{x_0}^{x_1} f(x)\cdot dx + \int_{x_1}^{x_2} f(x)\cdot dx + .. + \int_{x_i}^{x_{i+1}} f(x)\cdot dx + $$

$$+ .. + \int_{x_{N-1}}^{x_N} f(x)\cdot dx$$

Metodele prezentate în continuare aproximează mărimea unei suprafețe elementare prin diverse formule . Distanța între oricare două puncte alăturate este :

$$h = x_{i+1} - x_i = \frac{b - a}{N}$$

4) Metoda dreptunghiurilor

Se aproximează suprafața elementară cu cea a unui dreptunghi cu baza egală cu h și cu înălțimea egală cu $f(x_{i+1})$.

$$\int_{x_i}^{x_{i+1}} f(x)\cdot dx \approx f(x_{i+1})\cdot(x_{i+1} - x_i) = f(x_{i+1})\cdot h$$

$$I \approx h\cdot\Big[f(x_1) + f(x_2) + .. + f(x_N) \Big] \qquad (5)$$

5) Metoda trapezelor

Se aproximează suprafața elementară cu cea a unui trapez cu bazele egale cu $f(x_i)$ și $f(x_{i+1})$ și cu înălțimea egală cu h .

$$\int_{x_i}^{x_{i+1}} f(x)\cdot dx \approx \frac{f(x_i) + f(x_{i+1})}{2}\cdot(x_{i+1} - x_i) =$$

$$= \frac{h}{2}\cdot\Big[f(x_i) + f(x_{i+1}) \Big]$$

$$I \approx h\cdot\Big\{ \frac{f(x_0)}{2} + f(x_1) + f(x_2) + .. + f(x_{N-1}) + $$

$$+ \frac{f(x_N)}{2} \Big\} \qquad (6)$$

Această metodă este mai exactă decât metoda

dreptunghiurilor .

6) Metoda Simpson

Această metodă este cea mai exactă dintre metodele relativ simple , şi presupune un număr par de suprafeţe elementare (N par) .

Se aproximează suprafaţa a două suprafeţe elementare vecine cu cea limitată de o parabolă ce trece prin punctele :

$(x_i, f(x_i)), (x_{i+1}, f(x_{i+1})), (x_{i+2}, f(x_{i+2}))$

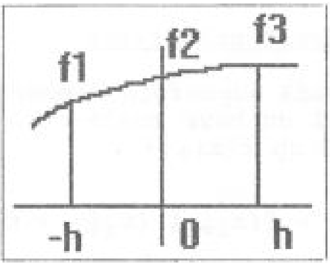
Avem $f(x) = a \cdot x^2 + b \cdot x + c$, iar condiţiile ca parabola să treacă prin cele trei puncte se scriu :

$a \cdot x_i^2 + b \cdot x_i + c = f(x_i)$

$a \cdot x_{i+1}^2 + b \cdot x_{i+1} + c = f(x_{i+1})$ cu $x_{i+1} = x_i + h$

$a \cdot x_{i+2}^2 + b \cdot x_{i+2} + c = f(x_{i+2})$ cu $x_{i+2} = x_i + 2 \cdot h$

Deoarece suprafaţa delimitată de parabolă nu depinde de alegerea originii pe axa Ox , ci numai de cele trei valori ale funcţiei în punctele separate prin h , trei puncte definind unic o parabolă , vom face calculul pentru următoarea configuraţie , rezultatul având însă valabilitate generală .



Condiţiile ca parabola să treacă prin cele trei puncte devin :

$a \cdot h^2 - b \cdot h + c = f1$

$c = f2$

$a \cdot h^2 + b \cdot h + c = f3$

Rezolvăm acest sistem de ecuaţii cu necunoscutele a , b şi c .

$a \cdot h^2 - b \cdot h = f1 - f2$

$a \cdot h^2 + b \cdot h = f3 - f2$

Prin sumarea şi scăderea acestor două ecuaţii obţinem :

$$a = \frac{f1 + f3 - 2 \cdot f2}{2 \cdot h^2} \qquad b = \frac{f3 - f1}{2 \cdot h} \qquad c = f2$$

$$\int_{-h}^{h} f(x) \cdot dx = \int_{-h}^{h} (a \cdot x^2 + b \cdot x + c) \cdot dx =$$

$$= \left[a\cdot\frac{x^3}{3} + b\cdot\frac{x^2}{2} + c\cdot x \right] \Bigg|_{-h}^{h} = \frac{a}{3}\cdot(h^3 - (-h)^3) +$$

$$+ \frac{b}{2}\cdot(h^2 - (-h)^2) + c\cdot(h - (-h)) =$$

$$= \frac{a}{3}\cdot 2\cdot h^3 + c\cdot 2\cdot h = \frac{f1 + f3 - 2\cdot f2}{2\cdot h^2}\cdot\frac{2\cdot h^3}{3} + f2\cdot 2\cdot h =$$

$$= \frac{h}{3}\cdot(f1 + f3 + 4\cdot f2)$$

Deci pentru cazul general avem :

$$\int_{x_i}^{x_{i+2}} f(x)\cdot dx \approx \frac{h}{3}\cdot\left[f(x_i) + 4\cdot f(x_{i+1}) + f(x_{i+2}) \right]$$

$$I \approx \frac{h}{3}\cdot\{ f(x_0) + 4\cdot f(x_1) + 2\cdot f(x_2) + 4\cdot f(x_3) +$$

$$+ 2\cdot f(x_4) + .. + 2\cdot f(x_{N-2}) + 4\cdot f(x_{N-1}) + f(x_N) \} \qquad (7)$$

7) Programul PASCAL

In continuare prezentăm un program ce calculează prin metoda trapezelor integrala ce dă probabilitatea ca o variabilă aleatoare supusă distribuţiei Gauss să fie la o distanţă mai mică decât z abateri standard de valoarea medie :

$$I = \int_{-z\cdot\sigma}^{z\cdot\sigma} \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left[\frac{-x^2}{2\cdot\sigma^2}\right]\cdot dx = \int_{-z}^{z} \frac{1}{\sqrt{2\pi}}\cdot\exp\left[\frac{-y^2}{2}\right]\cdot dy$$

Am utilizat schimbarea de variabilă : $y = \frac{x}{\sigma}$

Programul prezentat în continuare cere numărul de intervale în care este împărţit domeniul pe care se integrează , apoi efectuează integralele pentru $z \in \{ 1 , 2 , 3 , 4 \}$.

Integrarea este executată de către funcţia INTEGR , ce are ca parametru de intrare , pe lângă limitele domeniului pe care se integrează şi numărul de suprafeţe elementare în care acesta este împărţit , şi funcţia care este integrată , permiţând astfel efectuarea integrării pentru orice funcţie . Pentru a putea da unei funcţii ca parametru de intrare o altă funcţie , trebuie definit un tip de variabilă funcţie , iar programul trebuie compilat cu directiva Force Far Calls {$F+} .

```
{$F+}
PROGRAM INTEGRARE;
```

```
type fn=function(x:real):real;
var n,z:integer;

FUNCTION INTEGR(f:fn;a,b:real;n:integer):real;
{integrare prin metoda trapezelor}
var i:integer; dx,s:real;
begin
   dx:=(b-a)/n;
   s:=0; for i:=1 to n-1 do s:=s+f(a+i*dx);
   s:=s+f(a)/2+f(b)/2; integr:=s*dx;
end;

FUNCTION PG(x:real):real;
begin pg:=exp(-sqr(x)/2)/sqrt(2*pi); end;

begin
  writeln('Programul calculeaza probabilitatea ',
      'ca intr-o distributie Gauss ');
  writeln('eroarea sa nu fie mai mare de z ',
      'abateri standard'); writeln;
  writeln('nr de intervale pt calculul ',
      'integralei ( prin metoda trapezelor )');
  write('n = '); readln(n); writeln;
  for z:=1 to 4 do
     writeln('z= ',z,'      ',integr(pg,-z,z,n));
  readln;
end.
```

8) Probleme

1) Să se adapteze programul dat la metoda Simpson de integrare .

2) Se dau valorile $\{y_i\}$, $i=1,2,..,n$, ale unui spectru , corespunzătoare punctelor echidistante $\{x_i\}$, cu $x_i = x_1 + (i-1)\cdot h$, cu x_1 şi h date .

a) Folosind formula lui Simpson , să se calculeze suprafaţa acestui spectru .

b) Să se calculeze tabelul cu valorile derivatei de ordinul întâi şi de ordinul doi ale funcţiei y(x) , calculate numeric folosind valorile $\{y_i\}$.

c) Să se calculeze numărul picurilor de emisie şi numărul picurilor de absorbţie ale spectrului. Pic de emisie va fi considerat orice maxim , iar de absorbţie orice minim al funcţiei y(x) . Se va considera că între două puncte există un singur extrem dacă semnul derivatei întâi diferă în aceste puncte .

CAP V. INTEGRAREA PRIN METODA CUADRATURII GAUSS

1) Polinoame ortogonale

Ortogonalitatea a două polinoame este definită prin relația :

$$\int_a^b w(x)\cdot\phi_m(x)\cdot\phi_n(x) = \delta_{mn}\cdot C_m \qquad (1)$$

unde ϕ_m și ϕ_n sunt polinoame de gradul m , respectiv n , δ_{mn} este simbolul lui Kroneker , iar w(x) este o funcție pozitiv definită (adică nenegativă) numită pondere .

Pentru diferite valori a și b ale intervalului de integrare vom avea diferite seturi de polinoame ortogonale .

Cele mai folosite polinoame ortogonale sunt :

polinoamele Legendre

$a = -1$

$b = 1$

$w(x) = 1$

polinoamele Laguerre

$a = 0$

$b = \infty$

$w(x) = e^{-x}$

2) Ortogonalizarea prin metoda Gram-Schmidt

Construcția polinoamelor ortogonale se face simplu , pornind de la definiția (1) , polinoamele fiind deduse în ordinea crescătoare a gradului .

De multe ori , deși nu este obligatoriu , se impune și condiția de normare a polinoamelor :

$$C_m = 1 \qquad (2)$$

În acest caz formula (1) devine :

$$\int_a^b w(x)\cdot\phi_m(x)\cdot\phi_n(x) = \delta_{mn} \qquad (3)$$

Metoda Gram-Schmidt presupune pornirea de la un sistem de polinoame liniar independente , care de obicei este :

x^0 , x^1 , x^2 , x^3 , ...

Polinomul ortogonal de grad n este căutat ca o combinație liniară a polinoamelor ortogonale de grad inferior cu polinomul x^n , coeficienții combinației liniare fiind aflați în urma impunerii condițiilor (3) .

3) Metoda cuadraturii Gauss

Metodele de integrare numerică studiate până acum presupuneau împărțirea intervalului pe care

se făcea integrarea în N subintervale egale , și evaluarea valorilor funcției de integrat în cele N+1 puncte .

O mai bună precizie de calcul se obține dacă renunțăm la condiția restrictivă ca punctele în care evaluăm funcția să fie egal distanțate unele de altele .

Să considerăm o formulă de tipul :

$$\int_a^b f(x)\cdot w(x)\cdot dx \approx \sum_{m=1}^{N} W_m\cdot f(x_m) \qquad (4)$$

Avem ca necunoscute atât punctele x_m în care este evaluată funcția , cât și ponderile W_m cu care ele intervin în evaluarea integralei . In total avem 2N necunoscute .

Determinarea celor 2N necunoscute se face impunând ca , în cazul în care f(x) este un polinom oarecare de grad maxim 2N-1 , formula (4) să fie exactă (va rezulta un sistem de 2N ecuații corespunzătoare coeficienților puterilor lui x) . Acesta este principiul ce stă la baza integrării prin metoda cuadraturii Gauss .

Să considerăm f(x) polinom de gradul 2N-1 , și egalitatea :

$$\int_a^b f(x)\cdot w(x)\cdot dx = \sum_{m=1}^{N} W_m\cdot f(x_m) \qquad (5)$$

Prin împărțirea polinomului f(x) , de grad maxim 2N-1 , la $\phi_N(x)$, vom obține câtul $q_{N-1}(x)$ și restul $r_{N-1}(x)$, ambele de grad maxim N-1 .

$$f(x) = q_{N-1}(x)\cdot\phi_N(x) + r_{N-1}(x) \qquad (6)$$

Polinomul f(x) este ales arbitrar . Să-l alegem astfel încât :

$$r_{N-1}(x) = 0 \qquad (7)$$

Pe $q_{N-1}(x)$ îl putem dezvolta după sistemul de polinoame ortogonale :

$$q_{N-1}(x) = \sum_{i=0}^{N-1} q_i\cdot\phi_i(x) \qquad (8)$$

unde q_i sunt constante .

Folosind (6-8) , calculăm integrala din (5) :

$$\int_a^b f(x)\cdot w(x)\cdot dx = \int_a^b q_{N-1}(x)\cdot\phi_N(x)\cdot w(x)\cdot dx =$$

$$= \int_a^b \sum_{i=0}^{N-1} q_i\cdot\phi_i(x)\cdot\phi_N(x)\cdot w(x)\cdot dx =$$

$$= \sum_{i=0}^{N-1} q_i\cdot\int_a^b \phi_i(x)\cdot\phi_N(x)\cdot w(x)\cdot dx = 0$$

comform cu (3) , deoarece $i\neq N$.

Din (5-7) rezultă :

$$\sum_{m=1}^{N} W_m \cdot f(x_m) = \sum_{m=1}^{N} W_m \cdot q_{N-1}(x_m) \cdot \phi_N(x_m) = 0 \quad (9)$$

Deoarece polinomul f(x) este arbitrar , deci și polinomul q_{N-1} este arbitrar , rezultă că ecuația (9) este îndeplinită numai pentru :

$$\phi_N(x_m) = 0 \ , \ m=0..N-1 \quad (10)$$

Cele N valori ale lui x_m sunt tocmai cele N rădăcini ale polinomului ortogonal de gradul N . In cazurile de interes , aceste rădăcini sunt numere reale .

Mai rămâne să determinăm ponderile W_m .

Să definim următorul polinom de gradul N-1 , ce intervine și în polinomul Lagrange de interpolare :

$$l_{i,N}(x) = \frac{(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_N)}{(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_N)}$$

Acest polinom are interesanta proprietate că :

$$l_{i,N}(x_j) = \begin{cases} 0 \ , & j \neq i \\ 1 \ , & j = i \end{cases}$$

Acest polinom având gradul mai mic decât 2N-1 , trebuie să verifice egalitatea (5) :

$$\int_a^b l_{i,N}(x) \cdot w(x) \cdot dx = \sum_{m=1}^{N} W_m \cdot l_{i,N}(x_m) = W_i \quad (11)$$

Deci în felul acesta , prin calcularea analitică a integralei din (11) , se obțin ponderile .

4) <u>Scalarea domeniului de integrare</u>

In cazul în care a și b sunt finite , se vor folosi polinoamele Legendre .

Dacă $a \neq -1$ sau $b \neq 1$, se va face schimbarea de variabilă :

$$y = -1 + 2 \cdot \frac{x-a}{b-a} \quad (12)$$

astfel încât :

$$\int_a^b f(x) \cdot dx = \int_{-1}^{1} F(y) \cdot dy$$

In cazul în care $b=\infty$ se vor folosi polinoamele Laguerre .

Dacă $a \neq 0$ schimbarea de variabilă va fi :

$$y = x-a \quad (13)$$

astfel încât :

$$\int_a^{\infty} f(x)\cdot e^{-x}\cdot dx = \int_0^{\infty} F(y)\cdot e^{-y}\cdot dy$$

Mărirea preciziei cu care se estimează integrala se poate face prin creșterea numărului N de puncte , sau , mai eficient , prin împărțirea domeniului de integrare în mai multe subdomenii , pe fiecare subdomeniu integrala fiind calculată separat .

Concluzionând , formulele de integrare sunt :

$$\int_{-1}^{1} f(x)\cdot dx \approx \sum_{m=1}^{N} W_m\cdot f(x_m) \qquad \text{(Legendre)}$$

$$\int_{0}^{\infty} f(x)\cdot e^{-x}\cdot dx \approx \sum_{m=1}^{N} W_m\cdot f(x_m) \qquad \text{(Laguerre)}$$

Rădăcinile x_m și ponderile W_m sunt calculate și date în tabele , pentru economie de timp .

Cuadratura Gauss-Legendre
(rădăcinile și ponderile pentru N=10)

x_m	W_m
±0.973906528517172	0.066671344308688
±0.865063366688985	0.149451349150581
±0.679409568299024	0.219086362515982
±0.433395394129247	0.269266719309996
±0.148874338981631	0.295524224714753

Cuadratura Gauss-Laguerre
(rădăcinile și ponderile pentru N=10)

x_m	W_m
1.377934705404924E-1	3.084411157650201E-1
7.294545495031705E-1	4.011199291552736E-1
1.808342901740316	2.180682876118094E-1
3.401433697854900	6.208745609867775E-2
5.552496140063804	9.501506975181101E-3
8.330152746764497	7.530083885875388E-4
1.184378583790007E+1	2.825923349599566E-5
1.627925783137810E+1	4.249313984962686E-7
2.199658581198076E+1	1.048001174871510E-9
2.992069701227389E+1	9.911827219609009E-12

5) Programul PASCAL

In continuare se dă un program în PASCAL care calculează integrala :

$$I = \int_{-z}^{z} \frac{1}{\sqrt{2\pi}} \cdot \exp\left(\frac{-y^2}{2}\right) \cdot dy$$

Facem schimbarea de variabilă :

$y = x \cdot z$

$dy = z \cdot dx$

$$I = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} \cdot \exp\left(\frac{-x^2 \cdot z^2}{2}\right) \cdot z \cdot dx$$

Vom folosi integrarea prin metoda cuadraturii Gauss-Legendre , pentru funcția :

$$F(x) = \frac{z}{\sqrt{2\pi}} \cdot \exp\left(\frac{-x^2 \cdot z^2}{2}\right)$$

Rădăcinile și ponderile pentru polinomul Legendre de ordinul 10 sunt citite de către program din fișier , pentru că nu este practic să se repete pentru fiecare program scrierea celor 15 numere .

Iată programul :

```
{$F+}
PROGRAM INTGAUSS;
type fn=function(x:real):real;
var i,z:integer;
    x,w:array[1..10] of double;
    fx,fw:text;

FUNCTION INTG(F:fn):real;
var i:integer;
    s:double;
begin
     s:=0;
     for i:=1 to 10 do s:=s+w[i]*F(x[i]);
     intg:=s;
end;

FUNCTION F(x:real):real;
begin
     F:=z/sqrt(2*pi)*exp(-x*x*z*z/2);
end;

begin
    writeln('Integrare prin metoda cuadraturii');
    writeln('Gauss-Legendre cu N=10 puncte');
    writeln;
    writeln('Se calculeaza probabilitatea ca ',
            'intr-o distributie Gauss');
    writeln('eroarea sa nu fie mai mare de z ',
            'abateri standard');
    writeln;
    assign(fx,'legendre.x'); reset(fx);
    for i:=1 to 10 do readln(fx,x[i]);
    assign(fw,'legendre.w'); reset(fw);
```

```
    for i:=1 to 10 do readln(fw,w[i]);
    for i:=1 to 4 do begin
       z:=i;
       writeln('z= ',z,'     ',intg(F)); end;
    readln;
end.
```

6) Probleme

1) Deduceți polinoamele Legendre de gradul 0 , 1 și 2 .

Rezolvare :

$$\phi_0(x) = \alpha_{00}$$

$$m = 0\ ,\ n = 0\ ,\ (3) \Rightarrow \int_{-1}^{1} \alpha_{00}^2 \cdot dx = 1 \Rightarrow$$

$$\Rightarrow 2 \cdot \alpha_{00}^2 = 1 \Rightarrow \alpha_{00} = \frac{1}{\sqrt{2}}$$

$$\phi_0(x) = \frac{1}{\sqrt{2}} \qquad (0')$$

$$\phi_1(x) = \alpha_{11} \cdot x + \alpha_{10} \cdot \phi_0(x)$$

$$m = 1\ ,\ n = 0\ ,\ (3) \Rightarrow$$

$$\Rightarrow \int_{-1}^{1} (\alpha_{11} \cdot x + \frac{\alpha_{10}}{\sqrt{2}}) \cdot \frac{1}{\sqrt{2}} \cdot dx = 0 \Rightarrow \alpha_{10} = 0$$

$$m = 1\ ,\ n = 1\ ,\ (3) \Rightarrow \int_{-1}^{1} (\alpha_{11} \cdot x + \frac{\alpha_{10}}{\sqrt{2}})^2 \cdot dx = 1 \Rightarrow$$

$$\Rightarrow \frac{2}{3} \cdot \alpha_{11}^2 + \alpha_{10}^2 = 1 \Rightarrow \alpha_{11} = \sqrt{\frac{3}{2}}$$

$$\phi_1(x) = \sqrt{\frac{3}{2}} \cdot x \qquad (1')$$

$$\phi_2(x) = \alpha_{22} \cdot x^2 + \alpha_{21} \cdot \phi_1(x) + \alpha_{20} \cdot \phi_0(x)$$

$$m = 2\ ,\ n = 0\ ,\ (3) \Rightarrow$$

$$\Rightarrow \int_{-1}^{1} (\alpha_{22} \cdot x^2 + \alpha_{21} \cdot \sqrt{\frac{3}{2}} \cdot x + \alpha_{20} \cdot \frac{1}{\sqrt{2}}) \cdot \frac{1}{\sqrt{2}} \cdot dx = 0 \Rightarrow$$

$$\Rightarrow \alpha_{22} \cdot \frac{\sqrt{2}}{3} + \alpha_{20} = 0 \qquad (*)$$

$$m = 2\ ,\ n = 1\ ,\ (3) \Rightarrow$$

$$\Rightarrow \int_{-1}^{1} (\alpha_{22} \cdot x^2 + \alpha_{21} \cdot \sqrt{\frac{3}{2}} \cdot x + \alpha_{20} \cdot \frac{1}{\sqrt{2}}) \cdot \sqrt{\frac{3}{2}} \cdot x \cdot dx = 0 \Rightarrow$$

$$\Rightarrow \alpha_{21} = 0$$

m = 2 , n = 2 , (1) ⇒

$$\int_{-1}^{1} \left(\alpha_{22}\cdot x^2 + \alpha_{21}\cdot\sqrt{\frac{3}{2}}\cdot x + \alpha_{20}\cdot\frac{1}{\sqrt{2}}\right)^2\cdot dx = 0 \Rightarrow$$

$$\Rightarrow \alpha_{22}{}^2\cdot\frac{2}{5} + \alpha_{21}{}^2 + \alpha_{20}{}^2 + \sqrt{2}\cdot\alpha_{22}\cdot\alpha_{20}\cdot\frac{2}{3} = 1 \quad (**)$$

Rezolvând sistemul celor trei ecuații în necunoscutele α_{22} , α_{21} și α_{20} se obține :

$\alpha_{21} = 0$

$$(*) \Rightarrow \alpha_{20} = -\alpha_{22}\cdot\frac{\sqrt{2}}{3} \quad , \text{ introducem în } (**) \Rightarrow$$

$$\alpha_{22}{}^2\cdot\left(\frac{2}{5} + \frac{2}{9} - \frac{2\sqrt{2}}{3}\cdot\frac{\sqrt{2}}{3}\right) = 1 \Rightarrow \alpha_{22}{}^2\cdot\frac{8}{45} = 1 \Rightarrow$$

$$\alpha_{22} = \frac{3\sqrt{5}}{2\sqrt{2}}$$

$$\alpha_{20} = -\frac{\sqrt{5}}{2}$$

$$\phi_2(x) = \frac{\sqrt{5}}{2\sqrt{2}}\cdot(3\cdot x^2 - 1) \qquad (2')$$

Polinoamele au fost astfel definite până la un factor de semn arbitrar (dacă ϕ_n verifică (3) , atunci și $-\phi_n$ verifică (3)) .

2) Deduceți rădăcinile și ponderile pentru integrarea prin metoda cuadraturii Gauss-Legendre cu N=2 puncte .

Rezolvare :

$$\phi_2(x) = \frac{\sqrt{5}}{2\sqrt{2}}\cdot(3\cdot x^2 - 1) \Rightarrow x_1 = -\frac{1}{\sqrt{3}} \ , \ x_2 = +\frac{1}{\sqrt{3}}$$

$$W_1 = \int_{-1}^{1} l_{1,N}\cdot dx = \int_{-1}^{1} \frac{x-x_2}{x_1-x_2}\cdot dx = \frac{-2\cdot x_2}{x_1-x_2} = 1$$

$$W_2 = \int_{-1}^{1} l_{2,N}\cdot dx = \int_{-1}^{1} \frac{x-x_1}{x_2-x_1}\cdot dx = \frac{-2\cdot x_1}{x_2-x_1} = 1$$

CAP VI. INTEGRAREA ECUATIILOR DIFERENTIALE ORDINARE

1) Metoda Runge-Kutta de ordinul I (metoda Euler)

Se consideră ecuaţia diferenţială :

$$\frac{dy}{dx} = f(x,y) \qquad (1)$$

satisfăcând condiţia iniţială :

$$y(x_0) = y_0 \qquad (2)$$

$x = x_0 \Rightarrow y = y_0$

O dezvoltare în serie Taylor a funcţiei y(x) , în jurul punctului x_0 , duce la :

$$y(x_0+\Delta x) = y(x_0) + \frac{\Delta x}{1!}\cdot\frac{dy(x_0)}{dx} + \frac{\Delta x^2}{2!}\cdot\frac{d^2y(x_0)}{dx^2} + \\ + \dots \qquad (3)$$

Reţinând doar termenul de ordinul I, obţinem :

$$y(x_0+\Delta x) = y_0 + \Delta x\cdot\frac{dy(x_0)}{dx}$$

Limitarea la termenul de ordinul întâi al seriei este echivalentă cu aproximarea derivatei printr-un raport de diferenţe :

$$\frac{dy(x_0)}{dx} = \frac{y(x_0+\Delta x) - y_0}{\Delta x}$$

Folosind (1) se obţine :

$$y(x_0+\Delta x) = y_0 + \Delta x\cdot f(x_0,y_0) \qquad (4)$$

Procedeul se repetă , şi obţinem valorile lui y în punctele succesive separate prin Δx şi mai mari decât x_0 .

2) Metoda Runge-Kutta de ordinul II (metoda Euler modificata , imbunatatita)

După cum ne sugerează şi numele , va trebui într-o dezvoltare în serie Taylor să păstrăm termenii până în ordinul doi . Aceasta se face prin folosirea formulei de derivare numerică ce reţine termenul de ordinul doi al dezvoltării în serie Taylor . Punctul în care calculăm derivata se află la mijlocul intervalului $[x_0,x_0+\Delta x]$.

$$y'(x_0+\frac{\Delta x}{2}) = \frac{y(x_0+\Delta x) - y(x_0)}{\Delta x} \qquad (5)$$

In acelaşi timp :

$$y'(x_0+\frac{\Delta x}{2}) = f\left[x_0+\frac{\Delta x}{2},y\left[x_0+\frac{\Delta x}{2}\right]\right] \qquad (6)$$

Valoarea lui y o obţinem din prima metodă discutată :

$$y\left(x_0+\frac{\Delta x}{2}\right) = y_0 + \frac{\Delta x}{2}\cdot f(x_0,y_0) \qquad (7)$$

Din (5) , (6) şi (7) rezultă :

$$\frac{y(x_0+\Delta x) - y(x_0)}{\Delta x} = f\left[x_0+\frac{\Delta x}{2}, y_0 + \frac{\Delta x}{2}\cdot f(x_0,y_0)\right] \qquad (8)$$

De aici obţinem o formulă asemănătoare cu (4) , numai că derivata este calculată nu în x_0 , ci în punctul din mijlocul intervalului de interes $[x_0,x_0+\Delta x]$:

$$y(x_0+\Delta x) = y_0 + \Delta x\cdot f\left[x_0+\frac{\Delta x}{2}, y_0 + \frac{\Delta x}{2}\cdot f(x_0,y_0)\right] \qquad (9)$$

Aceasta este metoda lui Euler modificată . Metoda lui Euler îmbunătăţită o obţinem folosind, în loc de derivata în punctul x_0 , media derivatelor în punctele x_0 şi $x_0+\Delta x$.

Pentru metoda lui Euler îmbunătăţită vom avea:

$$y(x_0+\Delta x) = y_0 + \Delta x\cdot\frac{y'(x_0) + y'(x_0+\Delta x)}{2} \qquad (10)$$

$$y'(x_0+\Delta x) = f\left[x_0+\Delta x, y\left(x_0+\Delta x\right)\right] \qquad (11)$$

Valoarea lui y o obţinem din metoda de ordinul I :

$$y\left(x_0+\Delta x\right) = y_0 + \Delta x\cdot f(x_0,y_0) \qquad (12)$$

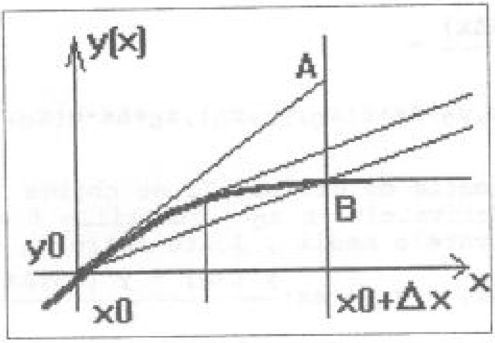
Din (10) , (11) şi (12) rezultă :

$$y(x_0+\Delta x) = y_0 + $$

$$+ \Delta x\cdot\frac{f(x_0,y_0) + f\left[x_0+\Delta x, y_0 + \Delta x\cdot f(x_0,y_0)\right]}{2} \qquad (13)$$

Aceasta este metoda Euler îmbunătăţită , sau metoda Runge-Kutta de ordinul II .

In ambele metode prezentate în acest subcapitol , faptul că ţinem cont că derivata y' nu este constantă în intervalul $[x_0,x_0+\Delta x]$, şi folosirea unor aproximaţii pentru derivata medie, ne duc la o soluţie mai bună decât cea dată de metoda de ordinul I . Pentru a înţelege cât mai bine acest lucru , prezentăm următorul grafic , în care se compară punctul A , obţinut prin metoda de ordinul I , cu punctul B , obţinut prin folosirea unei derivate medii pentru intervalul $[x_0,x_0+\Delta x]$. Se observă că mai aproape de graficul funcţiei y(x) este punctul B .



3) Metoda Runge-Kutta de ordinul IV

Aceasta este cea mai populară metodă de integrare , și constă în calculul unor cantități definite astfel :

$f_0 = f(x_0,y_0)$

$f_1 = f(x_0+\frac{\Delta x}{2},y_0+\frac{\Delta x}{2}\cdot f_0)$

$f_2 = f(x_0+\frac{\Delta x}{2},y_0+\frac{\Delta x}{2}\cdot f_1)$

$f_3 = f(x_0+\Delta x,y_0+\Delta x\cdot f_2)$

Cu aceste valori soluția se scrie :

$$f(x_0+\Delta x) = y_0 + \frac{\Delta x}{6}\cdot(f_0 + 2\cdot f_1 + 2\cdot f_2 + f_3)$$

4) Rezolvarea sistemelor de ecuatii diferentiale ordinare

Fie sistemul :

$$\frac{dy}{dx} = f(x,y,z) \qquad (14.1)$$

$$\frac{dz}{dx} = g(x,y,z) \qquad (14.2)$$

cu condițiile inițiale :

$$y(x_0) = y_0 \qquad (15.1)$$

$$z(x_0) = z_0 \qquad (15.2)$$

Presupunem că sunt satisfăcute condițiile de existență și unicitate ale soluției sistemului de ecuații diferențiale , pe un domeniu care include punctul (x_0,y_0,z_0) .

Aproximația de ordinul I ne dă :

$$y(x_0+\Delta x) = y_0 + \Delta x\cdot f(x_0,y_0,z_0) \qquad (16.1)$$

$$z(x_0+\Delta x) = z_0 + \Delta x\cdot g(x_0,y_0,z_0) \qquad (16.2)$$

Cu aceste valori calculăm derivatele funcțiilor în $x_0+\Delta x$:

$$\frac{dy(x_0+\Delta x)}{dx} =$$

$$= f\left[x_0+\Delta x,y_0+\Delta x\cdot f(x_0,y_0,z_0),z_0+\Delta x\cdot g(x_0,y_0,z_0)\right] \qquad (17.1)$$

$$\frac{dz(x_0+\Delta x)}{dx} =$$

$$= g\left[x_0+\Delta x, y_0+\Delta x\cdot f(x_0,y_0,z_0), z_0+\Delta x\cdot g(x_0,y_0,z_0)\right] \tag{17.2}$$

Aproximaţia de ordinul II se obţine înlocuind în (16) derivatele în x_0 (funcţiile f şi g în x_0) cu derivatele medii , luate intre x_0 şi $x_0+\Delta x$:

$$y(x_0+\Delta x) = y_0 + \Delta x\cdot\frac{y'(x_0) + y'(x_0+\Delta x)}{2} =$$

$$= y_0 + \frac{\Delta x}{2}\cdot\Big\{ f(x_0,y_0,z_0) +$$

$$+ f\left[x_0+\Delta x, y_0+\Delta x\cdot f(x_0,y_0,z_0), z_0+\Delta x\cdot g(x_0,y_0,z_0)\right] \Big\}$$

$$z(x_0+\Delta x) = z_0 + \Delta x\cdot\frac{z'(x_0) + z'(x_0+\Delta x)}{2} =$$

$$= z_0 + \frac{\Delta x}{2}\cdot\Big\{ g(x_0,y_0,z_0) +$$

$$+ g\left[x_0+\Delta x, y_0+\Delta x\cdot f(x_0,y_0,z_0), z_0+\Delta x\cdot g(x_0,y_0,z_0)\right] \Big\}$$

5) Programul PASCAL

In continuare prezentăm un program care rezolvă distribuţia de temperaturi nestaţionare într-un cilindru , starea iniţială fiind următoarea : cilindrul este descompus în doi cilindri coaxiali , cel interior ($r1 \le r \le r2$) şi cel exterior ($r2 \le r \le r3$) , fiecare având o anumită temperatură (de exemplu 0°C şi 100°C) .

Datorită simetriei cilindrice a problemei , ecuaţia ce caracterizează distribuţia de temperaturi se va scrie :

$$\frac{dT}{dt} = A\cdot\frac{d^2T}{dr^2} + B\cdot\frac{1}{r}\cdot\frac{dT}{dr} \tag{18}$$

unde r este coordonata radială , şi A este constanta termică a materialului . Aceasta este ecuaţia difuziei termice a lui Fourier pentru un cilindru infinit de lung .

Ecuaţia este rezolvată prin metoda Runge-Kutta de ordinul I , prin identificarea termenului drept din (18) cu o funcţie de T şi r . Valorile derivatelor vor fi calculate numeric , prin împărţirea intervalului [r1,r3] într-un şir de puncte echidistante , separate prin dr .

Programul presupune A=B=1 , şi cere diviziunile pe axa radială (dr) , şi pe axa timpului (dt) , precum şi valorile pentru r1 , r2 , r3 . Pentru fiecare moment de timp se calculează temperaturile în toate punctele diviziunii radiale . Aceste temperaturi se

reprezintă apoi grafic , dar nu pentru fiecare moment de timp , ci periodic , perioada (în unități dt) fiind introdusă tot de utilizatorul programului .

Derivatele în r1 și r3 (capetele intervalului) sunt calculate comform formulelor specifice , discutate la derivarea numerică .

Se va observa că pentru valori insuficient de mici pentru dr și dt rezultatul nu va fi corect , temperatura la extremitatea cilindrului rece crescând prea mult .

Ii este caracteristic metodei de ordinul I de a nu fi prea exactă , dar avantajele ei sunt ușurința cu care poate fi implementată , și suportul ei intuitiv imediat , metoda fiind foarte ușor de înțeles .

Iată în continuare programul :

```
PROGRAM ECDIF;
uses crt,graphics,graph;
var T,r,dT1,dT2:array[1..1000] of real;
    dt,dr,dr2:real; n,j:integer;
    i,n1,n2,n3:1..1000;
begin
  write('Pasul coordonatei radiale :  dr = ');
  readln(dr);
  writeln('Introduceti razele cilindrilor , ',
          'in unitati dr ( 1..1000 ) ');
  write('Raza mica a cil. interior :  n1 = ');
  readln(n1);
  write('Raza comuna a celor doi cil. :  n2 = ');
  readln(n2);
  write('Raza mare a cil. exterior :  n3 = ');
  readln(n3);
  if (n2<n1) or (n3<n1) or (n3<n2) then halt;
  dr2:=dr*dr; for i:=n1 to n3 do r[i]:=i*dr;
  write('Pasul de timp :  dt = '); readln(dt);
  write('Nr de pasi de timp pt desen :  n = ');
  readln(n);
  init_graph_mode;
  init_window(0,maxx,0,maxy,n1,n3,0,100);
  histo:=1;
  line(xe(n2),0,xe(n2),maxy);
  for i:=n1 to n2 do T[i]:=100;
  for i:=n2+1 to n3 do T[i]:=0;
 repeat
  for j:=1 to n do begin    { cei n pasi de timp
  }
    for i:=n1+1 to n3-1 do begin
       { prima derivata in i }
       dT1[i]:=(T[i+1]-T[i-1])/(2*dr);
       { a doua derivata in i }
       dT2[i]:=(T[i+1]+T[i-1]-2*T[i])/dr2; end;
    dT1[n1]:=(4*T[n1+1]-3*T[n1]-T[n1+2])/(2*dr);
    dT2[n1]:=(T[n1]+T[n1+2]-2*T[n1+1])/dr2;
    dT1[n3]:=(3*T[n3]+T[n3-2]-4*T[n3-1])/(2*dr);
```

```
dT2[n3]:=(T[n3]+T[n3-2]-2*T[n3-1])/dr2;
  for i:=n1 to n3 do
    T[i]:=T[i]+dt*(dT2[i]+dT1[i]/r[i]);
  end;
  for i:=n1 to n3 do punct_histo(i,T[i]);
until keypressed;
  closegraph;
end.
```

Vă sugerăm să încercați următorii parametrii de intrare :

```
dr = 0.1
n1 = 1
n2 = 80
n3 = 150
dt = 0.003
n = 100
```

CAP VII. REZOLVAREA ECUATIILOR CU DERIVATE PARTIALE PRIN METODA DIFERENTELOR FINITE

1) Introducere

Multe din ecuațiile de interes din fizică sunt ecuații cu derivate parțiale , adică implică două sau mai multe variabile independente (cum ar fi x și y) .

Metoda descrisă mai jos se poate adapta și la alte tipuri de ecuații , deși noi tratăm aici doar cazul ecuațiilor cu două variabile independente .

Noi vom studia distribuția de temperaturi staționare într-o placă dreptunghiulară , ce este guvernată de ecuația lui Laplace :

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \qquad (1)$$

Condițiile pe frontieră sunt de tip Dirichlet, adică temperaturile pe frontieră sunt fixate la niște valori date .

Metoda presupune următorii pași :

1) discretizarea ecuației cu derivate parțiale , cu ajutorul unei grile .

2) discretizarea condițiilor pe frontieră .

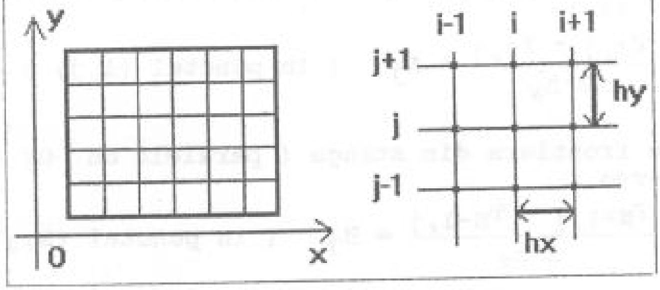
3) aflarea unor formule iterative pentru toate nodurile grilei .

4) repetarea iterației până când s-a atins precizia dorită sau până când s-a atins numărul maxim de iterații .

Deoarece rezultatul final , dat sub forma unui tabel cu temperaturile în nodurile grilei , este greu de urmărit , se folosește un algoritm ce trasează diferite izoterme , făcând rezultatul ușor accesibil ochiului .

2) Discretizarea ecuatiei

Domeniul este împărțit cu ajutorul unei grile, ce are pasul h_x pe direcția x , și pasul h_y pe direcția y .



Derivatele sunt calculate numeric folosind valorile funcției din nodurile grilei . Astfel , in punctul (x,y) , caracterizat de (i,j) , vom avea :

$$\frac{\partial T}{\partial x} \approx \frac{T(x+h_x,y) - T(x-h_x,y)}{2\cdot h_x} = \frac{T_{i+1,j} - T_{i-1,j}}{2\cdot h_x}$$

$$\frac{\partial T}{\partial y} \approx \frac{T(x,y+h_y) - T(x,y-h_y)}{2\cdot h_y} = \frac{T_{i,j+1} - T_{i,j-1}}{2\cdot h_y}$$

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T(x+h_x,y) + T(x-h_x,y) - 2\cdot T(x,y)}{{h_x}^2} =$$

$$= \frac{T_{i+1,j} + T_{i-1,j} - 2\cdot T_{i,j}}{{h_x}^2}$$

$$\frac{\partial^2 T}{\partial y^2} \approx \frac{T(x,y+h_y) + T(x,y-h_y) - 2\cdot T(x,y)}{{h_y}^2} =$$

$$= \frac{T_{i,j+1} + T_{i,j-1} - 2\cdot T_{i,j}}{{h_y}^2}$$

Ecuația (1) devine :

$$\frac{T_{i+1,j} + T_{i-1,j} - 2\cdot T_{i,j}}{{h_x}^2} +$$

$$+ \frac{T_{i,j+1} + T_{i,j-1} - 2\cdot T_{i,j}}{{h_y}^2} = 0 \quad (2)$$

3) <u>Discretizarea conditiilor pe frontiera</u>

a) Condiții de tip Dirichlet

Aceste condiții impun valori fixe ale funcției în punctele situate pe frontieră , adică în nodurile respective , și nu aduc nici o dificultate .

b) Condiții de tip Neumann

Aceste condiții impun valori fixe ale derivatei normale pe frontieră a funcției .

Presupunem i=1..N , j=1..M .

Pe frontiera din dreapta (paralelă cu Oy) vom avea :

$$\frac{\partial T}{\partial x} = \frac{T_{2,j} - T_{0,j}}{2\cdot h_x} = A_j \quad \text{(în punctul (1,j))}$$

(3)

Pe frontiera din stânga (paralelă cu Oy) vom avea :

$$\frac{\partial T}{\partial x} = \frac{T_{N+1,j} - T_{N-1,j}}{2\cdot h_x} = B_j \quad \text{(în punctul (N,j))}$$

(4)

Pe frontiera de jos (paralelă cu Ox) vom avea :

$$\frac{\partial T}{\partial y} = \frac{T_{i,2} - T_{i,0}}{2 \cdot h_y} = C_i \quad \text{(în punctul (i,1))} \tag{5}$$

Pe frontiera de sus (paralelă cu Ox) vom avea :

$$\frac{\partial T}{\partial y} = \frac{T_{i,M+1} - T_{i,M-1}}{2 \cdot h_y} = D_i \quad \text{(în punctul (i,M))} \tag{6}$$

Problema ce intervine în aceste puncte situate pe frontieră este că noi nu cunoaştem valorile funcţiei în puncte situate în exteriorul domeniului dreptunghiular studiat . Astfel , vom fi nevoiţi să combinăm ecuaţiile (3)-(6) cu ecuaţia (2) , pentru a elimina aceste necunoscute , ce corespund unor puncte fictive .

4) Aflarea unor formule iterative pentru toate nodurile grilei

Pentru punctele (i,j) situate în interiorul plăcii , din (2) obţinem :

$$T_{i,j} = \frac{h_x^2 \cdot h_y^2}{2 \cdot (h_x^2 + h_y^2)} \cdot \left(\frac{T_{i+1,j} + T_{i-1,j}}{h_x^2} + \frac{T_{i,j+1} + T_{i,j-1}}{h_y^2} \right) \tag{7}$$

In cazul în care $h_x = h_y$, formula (7) se simplifică la :

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4} \tag{8}$$

Pentru punctele de pe frontieră ce se supun condiţiilor de tip Dirichlet , valoarea funcţiei rămâne neschimbată .

Pentru punctele de pe frontieră ce se supun condiţiilor de tip Neumann , formula iteraţiei se deduce pentru fiecare latură în parte .

Pentru punctele aflate pe latura din dreapta , din (3) vom avea :

$$T_{0,j} = T_{2,j} - 2 \cdot h_x \cdot A_j \tag{9}$$

Punctul (0,j) este un punct fictiv .

Ecuaţia lui Laplace în punctul (1,j) se scrie:

$$\frac{T_{2,j} + T_{0,j} - 2 \cdot T_{1,j}}{h_x^2} + \frac{T_{1,j+1} + T_{1,j-1} - 2 \cdot T_{1,j}}{h_y^2} = 0 \tag{10}$$

Il scoatem pe $T_{0,j}$ din (9) , şi îl introducem

in (10) , obținându-l pe $T_{1,j}$ (mai ușor , il putem introduce pe $T_{0,j}$ direct in (7) , luând i=1) :

$$T_{1,j} = \frac{h_x^2 \cdot h_y^2}{2\cdot(h_x^2+h_y^2)} \cdot \left[\frac{T_{2,j} + T_{0,j}}{h_x^2} + \frac{T_{1,j+1} + T_{1,j-1}}{h_y^2} \right] =$$

$$= \frac{h_x^2 \cdot h_y^2}{2\cdot(h_x^2+h_y^2)} \cdot \left[\frac{2\cdot T_{2,j} - 2\cdot h_x \cdot A_j}{h_x^2} + \frac{T_{1,j+1} + T_{1,j-1}}{h_y^2} \right] \quad (11)$$

In cazul in care $h_x = h_y = h$, formula (11) se simplifică la :

$$T_{1,j} = \frac{2\cdot T_{2,j} - 2\cdot h\cdot A_j + T_{1,j+1} + T_{1,j-1}}{4} \quad (12)$$

In mod analog , pentru celelalte laturi se arată că :

$$T_{N,j} = \frac{2\cdot T_{N-1,j} + 2\cdot h\cdot B_j + T_{N,j+1} + T_{N,j-1}}{4} \quad (13)$$

$$T_{i,1} = \frac{T_{i+1,1} + T_{i-1,1} + 2\cdot T_{i,2} - 2\cdot h\cdot C_i}{4} \quad (14)$$

$$T_{i,M} = \frac{T_{i+1,M} + T_{i-1,M} + 2\cdot T_{i,M-1} + 2\cdot h\cdot D_i}{4} \quad (15)$$

O situație mai deosebită intâlnim la punctele situate chiar în colțuri . Aici trebuie verificate condițiile de derivată parțială fixă pe ambele axe , dar și ecuația lui Laplace conține două puncte fictive . Scoțând punctele fictive din condițiile de derivată parțială dată, și introducându-le în ecuația lui Laplace , obținem :

$$T_{1,1} = \frac{T_{1,2} - h\cdot C_1 + T_{2,1} - h\cdot A_1}{2} \quad (16)$$

$$T_{1,M} = \frac{T_{1,M-1} + h\cdot D_1 + T_{2,M} - h\cdot A_M}{2} \quad (17)$$

$$T_{N,1} = \frac{T_{N-1,1} + h\cdot B_1 + T_{N,2} - h\cdot C_N}{2} \quad (18)$$

$$T_{N,M} = \frac{T_{N-1,M} + h\cdot B_M + T_{N,M-1} + h\cdot D_N}{2} \quad (19)$$

5) Repetarea iteratiei

Metoda necesită două matrici , în care să fie stocate valorile funcției în punctele grilei . Se pornește cu o soluție aproximativă , introdusă în prima matrice . Cu ajutorul formulelor iterative menționate mai sus se calculează o aproximație mai bună , ce este stocată într-a doua matrice . După care rolul soluției aproximative inițiale este preluat de soluția aproximativă mai bună , iterația urmând să se repete până când soluția îmbunătățită nu diferă semnificativ de soluția precedentă . Aici se poate programa precizia cu care dorim să obținem soluția problemei . O măsură de siguranță este și impunerea ca numărul de iterații să nu depășească un număr maxim , deoarece există cazuri în care procesul nu converge către soluția căutată . Numărul de iterații total depinde de cât de bună este prima aproximație .

Nu trebuie pierdut din vedere faptul că ecuația cu diferențe finite nu este decât o aproximație a ecuației fizice date . De aceea este inutilă impunerea unei foarte mari precizii pentru condiția de ieșire din ciclul iterațiilor , fiindcă nu facem altceva decât să rezolvăm exact o ecuație care prin însăși natura ei e aproximativă . Dacă dorim o precizie mai bună , trebuie să mărim numărul de puncte din grilă , adică să micșorăm pașii h_x și h_y , realizând astfel o mai bună aproximație a ecuației fizice .

6) Metoda Supra Relaxarii Succesive

S-a observat că , în cazul unei bune alegeri a soluției aproximative inițiale , sau după câteve iterații , corecțiile făcute păstrează semn constant și sunt din ce în ce mai mici , pe măsură ce ne apropiem de soluția corectă . De aceea , pentru a grăbi convergența procesului iterativ , in loc de a inlocui valoarea veche ($Tv_{i,j}$) cu cea nouă ($Tn_{i,j}$) , rezultată în urma iterației în punctul respectiv , se preferă pentru noua valoare forma :

$$\underline{Tn_{i,j}} = Tn_{i,j} + \alpha \cdot (Tn_{i,j} - Tv_{i,j})$$

Această metodă poartă numele de Supra-Relaxare Succesivă (SOR , Successive Over-Relaxation) . Parametrul α se numește parametrul relaxării . Pentru rezolvarea unei ecuații în două variabile carteziene , ce este discretizată cu ajutorul aproximației de ordinul doi a dezvoltării centrale în serie Taylor (cea utilizată de noi) , valoarea optimă pentru α este :

$$\alpha = \frac{4}{2 + \sqrt{4 - \left[\cos(\frac{\pi}{N}) + \cos(\frac{\pi}{M})\right]^2}} - 1 \qquad (20)$$

unde N este numărul de puncte pe Ox și M este numărul de puncte pe Oy .

7) Programul PASCAL

In continuare este dat un program ce rezolvă distribuția de temperaturi staționare într-o placă dreptunghiulară . Condițiile pe frontieră sunt de tip Dirichlet , și anume : latura dreaptă și cea de jos sunt fixate la 0 grade , iar latura stângă și cea de sus sunt fixate la 100 de grade.

Deoarece în decursul unei iterații valorile funcției în nodurile grilei nu se transformă toate după aceeași formulă , ci s-a văzut că punctele de pe frontieră beneficiază de o mulțime de formule , este comod să introducem o matrice în care fiecărui punct să-i corespundă un indice (flag) al formulei care i se aplică în iterație . Acest lucru face mult mai comodă scrierea programului , chiar și când formulele iterative folosite sunt foarte numeroase (nu este cazul nostru , unde avem doar două categorii de puncte ,dar programul a fost scris exemplificativ,pentru a putea fi ușor adaptat și la alte cazuri) .

De asemenea , s-a preferat returnarea rezultatelor sub o formă grafică ușor de asimilat , prin trasarea unor izoterme . Algoritmul folosit analizează toată grila , pătrățel cu pătrățel , și fiecare pătrățel este analizat latură cu latură . Dacă se întâlnește o latură astfel încât valorile funcției în nodurile de la capetele ei să fie una mai mare , și cealaltă mai mică decât temperatura izotermei ce se trasează , atunci prin interpolare liniară se găsește un punct pe această latură care să aibă temperatura izotermei . Dacă pe laturile unui pătrățel se găsesc două astfel de puncte , atunci între ele se traseaza un segment de dreaptă . Deoarece pătrățelele sunt foarte mici , curba astfel trasată din segmente de dreaptă aproximează destul de bine izoterma .

```
program diferente_finite;
{ rezolva ecuatia lui Laplace pentru distributia
de temperaturi stationare dintr-o placa
dreptunghiulara care are doua laturi fixate
la 100 grC , si celelalte doua fixate la 0 grC }
uses crt,graph,graphics;
const errel=1E-3;     { eroarea relativa maxima }
      itmax=1000;          { nr maxim de iteratii }
var pv,pn:array[1..60,1..60] of real;
```

```
    pf:array[1..60,1..60] of integer;
    nx,ny,i,j,it:integer;
    hx,hy:real;
    gata:boolean;

{ CONDITII PE FRONTIERA DE TIP DIRICHLET }
function f1(i,j:integer):real;
begin
     f1:=pv[i,j];
end;

{ ECUATIE LAPLACE }
function f2(i,j:integer):real;
begin
     f2:=hx*hx*hy*hy/2/(hx*hx+hy*hy)*
         ( (pv[i+1,j]+pv[i-1,j])/(hx*hx) +
           (pv[i,j+1]+pv[i,j-1])/(hy*hy) );
end;

{ REPREZENTARE GRAFICA A UNEI IZOTERME }
procedure izo(t:real);
var x,y:array[1..4] of real;
    i,j,n:integer;
begin
for i:=1 to nx-1 do
for j:=1 to ny-1 do
  begin
  n:=0;
  if (pn[i,j]-t)*(pn[i+1,j]-t)<=0 then begin
     n:=n+1;
     x[n]:=i+(t-pn[i,j])/(pn[i+1,j]-pn[i,j]);
     y[n]:=j; end;
  if (pn[i+1,j]-t)*(pn[i+1,j+1]-t)<=0 then begin
     n:=n+1;
     x[n]:=i+1;
     y[n]:=j+(t-pn[i+1,j])/
                         (pn[i+1,j+1]-pn[i+1,j]); end;
  if (pn[i,j+1]-t)*(pn[i+1,j+1]-t)<=0 then begin
     n:=n+1;
     x[n]:=i+(t-pn[i,j+1])/
                              (pn[i+1,j+1]-pn[i,j+1]);
     y[n]:=j+1; end;
  if (pn[i,j]-t)*(pn[i,j+1]-t)<=0 then begin
     n:=n+1;
     x[n]:=i;
     y[n]:=j+(t-pn[i,j])/(pn[i,j+1]-pn[i,j]);end;
  if n=2
  then line(xe(x[1]),ye(y[1]),xe(x[2]),ye(y[2]));
  end;
end;

{ INCEPUTUL PROGRAMULUI }
begin
     nx:=50; ny:=50; hx:=1; hy:=1;

     for i:=1 to nx do
     begin pv[i,1]:=0; pf[i,1]:=1; end;
```

```
    for j:=1 to ny do
    begin pv[1,j]:=0; pf[1,j]:=1; end;
    for i:=2 to nx do
    begin pv[i,ny]:=100; pf[i,ny]:=1; end;
    for j:=2 to ny do
    begin pv[nx,j]:=100; pf[nx,j]:=1; end;
    for i:=2 to nx-1 do
    for j:=2 to ny-1 do
    begin pv[i,j]:=50; pf[i,j]:=2; end;

    it:=0;
repeat

{ ITERATIA }
    for i:=1 to nx do
    for j:=1 to ny do
       case pf[i,j] of
          1: pn[i,j]:=f1(i,j);
          2: pn[i,j]:=f2(i,j);
       end;

{ S-A ATINS PRECIZIA DORITA ? }
    gata:=true;
    for i:=1 to nx do
    for j:=1 to ny do
       if pf[i,j]<>1 then
         if abs((pn[i,j]-pv[i,j])/pn[i,j])>errel
         then gata:=false;

{ S-A DEPASIT NR MAXIM DE ITERATII ? }
    it:=it+1;
    if it>itmax then gata:=true;

{ INNOIREA MATRICII CU PUNCTELE }
    for i:=1 to nx do
    for j:=1 to ny do
       pv[i,j]:=pn[i,j];

until gata;

init_graph_mode;
init_window(0,maxy,0,maxy,1,nx,1,ny);
for i:=1 to 9 do izo(10*i);
repeat until readkey='q';
closegraph;

end.
```

8) Problema

Modificați valorile temperaturilor de pe frontiere , și urmăriți cum se schimbă forma izotermelor . Studiați cazul în care pe latura stângă și pe cea de jos temperatura variază liniar de la 0 la 100 , având 0 în colțul din stânga-jos , iar pe celelalte două laturi avem în continuare tot 100 de grade .

CAP VIII. SISTEME DE ECUATII LINIARE SI NELINIARE

1) Aducerea unei matrici la forma superior triunghiulara

Aducerea unei matrici la forma superior triunghiulară , adică la forma în care sub diagonală avem zero , prin operații efectuate asupra liniilor și coloanelor ei , este o operație comună algoritmilor de rezolvare a următoarelor probleme : rezolvarea unui sistem de ecuații liniare prin metoda Gauss , calculul determinantului unei matrici , calculul inversei unei matrici .

Fie A o matrice pătratică de dimensiune n , cu elementele a(i,j) , i,j=1..n . Notăm liniile ei cu L(i) și coloanele cu C(j) . Operațiile pe care le vom face sunt :

1) schimbarea între ele a două linii :
L(i) ⇔ L(j)

2) schimbarea între ele a două coloane :
C(i) ⇔ C(j)

3) scăderea unei linii , înmulțită cu o constantă , dintr-o altă linie :
L(i) ⇐ L(i) - t·L(j)

Prelucrăm matricea , modificând coloanele , începând cu C(1) și sfârșind cu C(n) , astfel încât în coloana C(k) , k=1..n , sub elementul diagonal a(k,k) să obținem numai elemente nule : a(i,k)=0 pentru i=k+1..n . Pentru un i dat , elementul a(i,k) ajunge zero în urma operației următoare :

L(i) ⇐ L(i) - t·L(k)

unde t=a(i,k)/a(k,k) , adică :

```
t:=a[i,k]/a[k,k];
for j:=1 to n do a[i,j]:=a[i,j]-t*a[k,j];
```

Variabila t depinde de i și de k , ea fiind definită în interiorul ciclului după i , care la rândul lui este în interiorul ciclului după k . Deoarece în momentul prelucrării coloanei k , coloanele din stânga ei (adică C(1)..C(k-1)) au realizat deja zero sub elementul diagonal înseamnă că putem restrânge domeniul lui j numai la elementele nenule , adică j=k..n .

Pentru a îl calcula pe t , trebuie să ne asigurăm că a(k,k)<>0 . In cazul nefavorabil va trebui să căutăm , în submatricea a(i,j) cu i,j=k..n , un alt element nenul a(l1,l2) , având k≤l1,l2≤n , iar apoi să schimbăm între ele liniile și coloanele :

L(k) ⇔ L(l1)   și   C(k) ⇔ C(l2)

In felul acesta a(l1,l2) va lua locul lui a(k,k) . Datorită faptului că elementul nenul a(l1,l2) a fost căutat în submatricea a(i,j) cu i,j=k..n , schimbarea de linii și de coloane

realizată nu afectează zero-ul realizat sub diagonală pentru coloanele C(1) .. C(k-1) .

Dacă nu se găsește nici un element nenul , programul se oprește .

In practică , chiar dacă a(k,k)<>0 , noi vom căuta în submatrice elementul maxim în valoare absolută (pivotul) , și vom face schimbarea de linii și de coloane , pentru a avea erori cât mai mici în calculul lui t .

2) Calcularea determinantului unei matrici

Folosim faptul că determinantul unei matrici aduse la forma superior triunghiulară este egal cu produsul elementelor diagonale . Va trebui doar să ținem cont de modificările aduse determinantului de către operațiile pe care le facem noi asupra matricii , adică :

1) L(k) ⇔ L(l1)   produce schimbarea semnului
2) C(k) ⇔ C(l2)   produce schimbarea semnului
3) L(i) ⇐ L(i) - t·L(k) nu modifică determinantul

Vom introduce o variabilă suplimentară d în care vom păstra semnul determinantului . Inițial d:=1 . La schimbarea semnului vom executa și d:=-d . După ce matricea a fost adusă la forma dorită :

determinantul = d·a(1,1)·a(2,2)·...·a(n,n)

In practică vom calcula acest produs în paralel cu prelucrarea matricii , în cadrul ciclului după k .

3) Metoda Gauss pentru rezolvarea sistemelor liniare de ecuatii

Metoda Gauss este o metodă exactă de rezolvare a unui sistem de n ecuații cu n necunoscute . Fie un sistem de ecuații dat sub forma :

a(1,1)·x(1) + a(1,2)·x(2) +..+ a(1,n)·x(n) = b(1)
a(2,1)·x(1) + a(2,2)·x(2) +..+ a(2,n)·x(n) = b(2)
....
a(n,1)·x(1) + a(n,2)·x(2) +..+ a(n,n)·x(n) = b(n)

Efectuând asupra lui operațiile ce urmăresc anularea coeficienților de sub diagonală , va trebui să modificăm și matricile B și X , astfel încât soluția să se păstreze . Operațiile permise sunt :

1) schimbarea ordinii de numerotare a ecuațiilor :

L(i) ⇔ L(j)   și   b(i) ⇔ b(j)

2) schimbarea ordinii de numerotare a necunoscutelor :

x(i) ⇔ x(j) , C(i) ⇔ C(j) și p(i) ⇔ p(j)

unde P este o matrice în care păstrăm vechea ordine de numerotare a necunoscutelor , inițial având p(i)=i .

3) scăderea unei ecuații , înmulțite cu o constantă , dintr-o altă ecuație :

$L(i) \leftarrow L(i) - t \cdot L(k)$ și $b(i) \leftarrow b(i) - t \cdot b(k)$

După modificările aduse,sistemul va arăta așa:

$$
\begin{array}{lllll}
a(1,1)\cdot x(1) & + a(1,2)\cdot x(2) & +..+ & a(1,n)\cdot x(n) & = b(1) \\
0 & a(2,2)\cdot x(2) & +..+ & a(2,n)\cdot x(n) & = b(2) \\
\dots & \dots & \dots & \dots & \dots \\
0 & 0 & +..+ & a(n,n)\cdot x(n) & = b(n)
\end{array}
$$

Sub această formă sistemul se rezolvă ușor , calculând valoarea lui x(n) , apoi cea a lui x(n-1) , și tot așa până la x(1) .

$$x(n) = \frac{b(n)}{a(n,n)}$$

$$x(n-1) = \frac{b(n-1) - a(n-1,n)\cdot x(n)}{a(n-1,n-1)}$$

$$x(i) = \frac{b(i) - \sum_{j=i+1}^{n} a(i,j)\cdot x(j)}{a(i,i)}$$

La returnarea rezultatelor va trebui să ținem cont de modificările aduse ordinii de numerotare a necunoscutelor . Astfel x(i) nu va conține valoare necunoscutei i , ci cea a necunoscutei al cărei număr e stocat în p(i) .

4) Inversarea unei matrici

Fie C matricea inversă matricii pătratice date A , iar E matricea unitate de același rang n . Considerând elementele lui C necunoscute , avem condiția ca AC=E , adică :

$$\sum_{j=1}^{n} a(i,j)\cdot c(j,l) = e(i,l) = \delta il$$

pentru $\forall i=1..n$, $\forall l=1..n$

Simbolul lui Kroneker este definit :

$$\delta il = \begin{cases} 1 \ , \ i=l \\ 0 \ , \ i \neq l \end{cases}$$

Aceasta ne conduce la n sisteme de ecuații liniare cu câte n necunoscute . Variația lui l ne dă diferitele sisteme de ecuații , iar variația lui i ne dă diferitele ecuații ale unui sistem . Pentru fiecare valoare a lui l avem câte un sistem cu termenii liberi egali cu coloana l a matricii E , necunoscutele fiind elementele coloanei l a matricii C , toate sistemele având aceeași matrice principală A . Aceasta inseamnă că aducerea matricii A la forma superior triunghiulară intervine în rezolvarea simultană a celor n sisteme pentru determinarea elementelor matricii inverse C . Cele n operații asupra

termenilor liberi , făcute pentru cele n sisteme de ecuații (l=1..n) , vor realiza de fapt o operație asupra liniilor L_E ale matricii E :

$e(k,l) \Leftrightarrow e(l1,l) \quad \Rightarrow \quad L_E(k) \Leftrightarrow L_E(l1)$

$e(i,l) \leftarrow e(i,l) - t \cdot e(k,l) \quad \Rightarrow$

$\Rightarrow \quad L_E(i) \leftarrow L_E(i) - t \cdot L_E(k)$

Cele n operații de schimbare a ordinii necunoscutelor , făcute pentru cele n sisteme , vor realiza de fapt o schimbare a liniilor L_C ale matricii C :

$c(k,l) \Leftrightarrow c(l2,l) \quad \Rightarrow \quad L_C(k) \Leftrightarrow L_C(l2)$

Ordinea inițială a liniilor va fi restabilită cu ajutorul matricii P în care am păstrat permutările făcute .

5) Rezolvarea sistemelor de ecuatii neliniare

Fie $F_i(x_1,..,x_n) = 0$, i=1..n , un sistem de n ecuații cu n necunoscute ,cu o soluție $(\bar{x}_1,..,\bar{x}_n)$. Notăm $x=(x_1,..,x_n)$ și $\bar{x}=(\bar{x}_1,..,\bar{x}_n)$.

Presupunem că funcțiile $F_i(x)$ și $\dfrac{\partial F_i(x)}{\partial x_k}$ sunt

continue , și că determinantul $\left|\dfrac{\partial F_i(x)}{\partial x_k}\right|$ este

nenul . Se poate arăta că în aceste condiții , pornind de la o soluție aproximativă $x^0=(x_1^0,...,x_n^0)$, se poate să se ajungă oricât de aproape de soluția exactă $\bar{x}$, prin oricare din metodele următoare :

6) Metoda Newton

Dezvoltând funcțiile $F_i(\bar{x})$ în serie Taylor în jurul punctului x^0 și neglijând termenii de ordin superior lui unu , obținem :

$$F_i(\bar{x}) = F_i(x^0) + \sum_{k=1}^{n} \frac{\partial F_i(x^0)}{\partial x_k} \cdot (\bar{x}_k - x_k^0) = 0 \quad , i=1..n$$

Notăm $\Delta x_k^1 = \bar{x}_k - x_k^0$. Am obținut un sistem de n ecuații liniare, cu necunoscutele Δx_k^1 . Din cauză că liniarizarea sistemului s-a făcut prin neglijarea unor termeni de ordin superior din dezvoltarea în serie Taylor , soluția Δx_k^1 va fi numai aproximativ egală cu diferența dintre soluția exactă $\bar{x}_k$ și cea aproximativă x_k^0 . Totuși , dacă x^0 este suficient de aproape de $\bar{x}$, aproximația următoare $x^1 = x^0 + \Delta x^1$ va fi mai aproape de $\bar{x}$, decât aproximația inițială x^0 .

Repetând acest proces se obține un șir de

soluții $\{x^l\}$ generat de formula de recurență

$x^l = x^{l-1} + \Delta x^l$, l=1,2,3,... în care Δx^l este soluția sistemului de ecuații liniare , în care soluția aproximativă este luată x^{l-1} .

Ieșirea din ciclul iterațiilor se face atunci

când s-a atins precizia dorită ($\max\left\{abs\left(\frac{\Delta x_k^l}{x_k^l}\right)\right\}<ER$

, pentru eroarea relativă maximă ER , sau

$\max\left\{abs(\Delta x_k^l)\right\}<EA$, pentru eroarea absolută maximă

EA) sau când s-a atins numărul maxim de iterații imax (l=imax) .

7) Metoda Newton-Gauss

Este analoagă cu metoda Newton , însă introduce suplimentar verificarea convergenței șirului $\{x^l\}$ spre soluția exactă $\bar{x}$, prin minimizarea funcției :

$$\chi^2(x) = \sum_{i=1}^{n} [F_i(x)]^2$$

Cu cât $\chi^2(x)$ este mai mic , cu atât x e mai aproape de $\bar{x}$. In cazul în care $\chi^2(x^l)>\chi^2(x^{l-1})$ corecția Δx^l se micșorează , până când se ajunge la $\chi^2(x^l)<\chi^2(x^{l-1})$.

8) Metoda gradientului

Se știe că gradientul unei funcții indică direcția și sensul creșterii maxime . Noi vom căuta o corecție pentru a avea scăderea maximă a lui $\chi^2(x^l)$, deci pe aceeași direcție , dar de sens opus gradientului :

$$\Delta x^l = (\Delta x_1^l, \Delta x_2^l, \ldots, \Delta x_n^l) =$$

$$= K\cdot\left(\frac{\partial\chi^2(x^l)}{\partial x_1}, \frac{\partial\chi^2(x^l)}{\partial x_2}, \ldots, \frac{\partial\chi^2(x^l)}{\partial x_n}\right)$$

Factorul de proporționalitate K este negativ (pentru a indica sensul opus gradientului), și îl vom obține din dezvoltarea în serie Taylor a funcției $\chi^2(\bar{x})$ în jurul lui x^l , reținând doar termenii de ordinul întâi :

$$0 = \chi^2(\bar{x}) \approx \chi^2(x^l+\Delta x^l) \approx \chi^2(x^l) + \sum_{i=1}^{n} \frac{\partial\chi^2(x^l)}{\partial x_i}\cdot\Delta x_i^l$$

$$\Rightarrow \chi^2(x^l) + \sum_{i=1}^{n} \left(\frac{\partial\chi^2(x^l)}{\partial x_i}\right)^2 \cdot K = 0 \Rightarrow$$

$$\Rightarrow K = - \frac{\chi^2(x^l)}{\sum_{i=1}^{n} \left(\frac{\partial\chi^2(x^l)}{\partial x_i}\right)^2}$$

In concluzie , formula de recurență este :

$$x_k^{l+1} = x_k^l - \frac{\chi^2(x^l)}{\sum_{i=1}^{n} \left(\frac{\partial\chi^2(x^l)}{\partial x_i}\right)^2} \cdot \frac{\partial\chi^2(x^l)}{\partial x_k} \qquad k=1..n$$

sau scrisă vectorial :

$$x^{l+1} = x^l - \frac{\chi^2(x^l)}{\sum_{i=1}^{n} \left(\frac{\partial\chi^2(x^l)}{\partial x_i}\right)^2} \cdot \mathrm{grad}(\chi^2(x^l))$$

Pentru a mări și mai mult viteza de convergență a iterațiilor , amplitudinea corecției (K) poate fi suplimentar ajustată .

9) Programele in PASCAL

In continuare prezentăm două programe , unul care rezolvă prin metoda Gauss un sistem de ecuații liniare , și unul care rezolvă prin metoda Newton-Gauss un sistem de trei ecuații neliniare . Programele siseclin.pas și sisecnel.pas conțin introducerea datelor și scoaterea rezultatelor , subrutinele de calcul fiind scrise în unitul sisec.pas pentru a putea fi ușor folosite și de alte programe . Pentru sistemul de ecuații liniare , se vor introduce de la tastatură coeficienții ecuațiilor , iar pentru cel al ecuațiilor neliniare funcțiile sunt date în program , împreună cu derivatele lor parțiale (în cazul nostru calculate numeric) , și de la tastatură se va introduce doar soluția aproximativă inițială (încercați x1=-2, x2=1 , x3=-2) .

Pentru a asigura convergența procesului de iterație , metoda Newton-Gauss a fost modificată în sensul introducerii unui parametru g în relația : $x^l = x^{l-1} + g \cdot \Delta x^l$, care face ca funcția χ^2 să depindă de acest parametru : $\chi^2 = \chi^2(g)$. Parametrul g se alege astfel încât parabola care trece prin punctele $(0,\chi^2(0))$, $(0.5,\chi^2(0.5))$, $(1,\chi^2(1))$ să admită o valoare minimă tocmai pentru această valoare g .

$$\chi^2(x) = a \cdot x^2 + b \cdot x + c$$

$$g = -\frac{b}{2\cdot a} = \frac{\chi^2(1)+3\chi^2(0)-4\chi^2(0.5)}{4(\chi^2(1)+\chi^2(0)-2\chi^2(0.5))}$$

Iată în continuare programele :

```
UNIT SISEC;

interface

type dec=1..10;
     mp=array[dec,dec] of real;
     mc=array[dec] of real;
     fnc=function(i:dec;x:mc):real;
     dfnc=function(i,j:dec;x:mc):real;
procedure sismat(n:dec;a:mp;b:mc;
                  var det:real;var s:mc;var r:mp);
procedure sisiter(n:dec;f:fnc;df:dfnc;
                  var x0:mc;var iter:integer);
function maxim(n:dec;x:mc):real;

implementation

PROCEDURE SISMAT;
var d,max,t:real; c,e:mp; x:mc;
    p:array[dec] of dec; i,j,k,l1,l2:dec;
begin
for i:=1 to n do begin {initializarile pt E si P}
   for j:=1 to n do if i=j then e[i,j]:=1
                            else e[i,j]:=0;
   p[i]:=i; end;
d:=1;
for k:=1 to n do             {DESCHIDE CICLU DUPA k}
begin
  max:=abs(a[k,k]);           {cauta pivotul maxim}
  l1:=k; l2:=k;
  for i:=k to n do
  for j:=k to n do
  if abs(a[i,j])>max then
    begin max:=abs(a[i,j]); l1:=i; l2:=j; end;
  if max=0.0 then begin det:=0; exit; end;
  {determinant nul ?}
  if l1<>k then begin   {schimbare de linii in A}
    d:=-d;                {semnul determinantului}
    t:=b[k]; b[k]:=b[l1]; b[l1]:=t;
    for i:=1 to n do
      begin
      t:=e[k,i]; e[k,i]:=e[l1,i]; e[l1,i]:=t;
      end;
    for i:=k to n do
      begin
      t:=a[k,i]; a[k,i]:=a[l1,i]; a[l1,i]:=t;
      end;
  end;
  if l2<>k then begin {schimbare de coloane in A}
    d:=-d;                {semnul determinantului}
    j:=p[k]; p[k]:=p[l2]; p[l2]:=j;
    for i:=1 to n do
```

```
        begin
        t:=a[i,k]; a[i,k]:=a[i,l2]; a[i,l2]:=t;
        end;
   end;
   for i:=k+1 to n do        {DESCHIDE CICLU DUPA i}
   begin          {scadere de linii in matricea A}
        t:=a[i,k]/a[k,k];
        b[i]:=b[i]-b[k]*t;
        for j:=k to n do a[i,j]:=a[i,j]-a[k,j]*t;
        for j:=1 to n do e[i,j]:=e[i,j]-e[k,j]*t;
   end;                         {INCHIDE CICLU DUPA i}
   d:=d*a[k,k];                   {calcul determinant}
end;                            {INCHIDE CICLU DUPA k}
det:=d;
for i:=n downto 1 do begin
    t:=0;
    for j:=i+1 to n do t:=t+a[i,j]*x[j];
    x[i]:=(b[i]-t)/a[i,i];                 {calcul X}
    for k:=1 to n do begin  {n sisteme de ec pt C}
       t:=0;
       for j:=i+1 to n do t:=t+a[i,j]*c[j,k];
       c[i,k]:=(e[i,k]-t)/a[i,i];          {calcul C}
    end;
end;
for i:=1 to n do begin
{revenirea la ordinea initiala de numerotare}
    s[p[i]]:=x[i];
    for k:=1 to n do r[p[i],k]:=c[i,k]; end;
end;

function hi2(n:dec;f:fnc;x:mc):real;
var h:real; i:integer;
begin
     h:=0;
     for i:=1 to n do h:=h+sqr(f(i,x));
     hi2:=h;
end;

FUNCTION MAXIM;
var m:real; i:integer;
begin
     m:=x[1];
     for i:=2 to n do if m<x[i] then m:=x[i];
     maxim:=m;
end;

PROCEDURE SISITER;
var a,c:mp; dx,b,x1,dxr:mc; i,j:integer;
    det,h0,h05,h1,g:real;
begin
     iter:=0;
repeat
  for i:=1 to n do begin
  {sistemul de ecuatii liniare rezultat din
  dezvoltarea in serie Taylor a functiilor}
     b[i]:=-f(i,x0);
     for j:=1 to n do a[i,j]:=df(i,j,x0); end;
```

```
   sismat(n,a,b,det,dx,c);
   if det=0 then begin iter:=101; exit; end;
   h0:=hi2(n,f,x0);
   for i:=1 to n do x1[i]:=x0[i]+0.5*dx[i];
   h05:=hi2(n,f,x1);
   for i:=1 to n do x1[i]:=x0[i]+dx[i];
   h1:=hi2(n,f,x1);
   g:=(h1+3*h0-4*h05)/(4*(h1+h0-2*h05));
   if (g<0) or (g>1.5) then g:=1;
   for i:=1 to n do  {calculul corectiei relative}
   begin
     dxr[i]:=g*abs(dx[i])/x0[i];
     x0[i]:=x0[i]+g*dx[i];
   end;
   iter:=iter+1;
until (maxim(n,dxr)<0.000001) or (iter=100);
end;

end.


PROGRAM SISECLIN;
uses sisec;
var a,c:mp; b,x:mc; n,i,j:dec; det:real;

begin
write('ordinul matricii (1..10)    n=');
readln(n);
for i:=1 to n do begin
for j:=1 to n do begin
   write('a(',i,',',j,')= ');
   readln(a[i,j]); end;      {introducere A}
   write('b(',i,')= ');
   readln(b[i]); end;        {introducere B}
sismat(n,a,b,det,x,c);
writeln; writeln('det A = ',det);
if det=0 then begin readln; halt; end;
writeln; writeln('solutia sistemului');
for i:=1 to n do writeln('x(',i,')= ',x[i]);
writeln;
writeln('matricea inversa     <ENTER> pt linie');
for i:=1 to n do begin
   writeln('linia ',i);
   for j:=1 to n do
     writeln('c(',i,',',j,')= ',c[i,j]);
   readln; end;
end.


{$F+}
PROGRAM SISECNEL;
uses sisec;
var x:mc; i,iter:integer;

FUNCTION F(i:dec;x:mc):real;
begin
     case i of
```

```
    1: f:=x[1]+sqr(x[2])+x[3];
    2: f:=sqr(x[1])+sqr(x[2])+sqr(x[3])-4;
    3: f:=3*x[1]+x[2]+x[3]+4-sqrt(2);
    end;
end;

FUNCTION DF(i,j:dec;x:mc):real;
const delta:real=0.0001;
var fm,fp:real;
begin
    x[j]:=x[j]-delta; fm:=f(i,x);
    x[j]:=x[j]+2*delta; fp:=f(i,x);
    df:=(fp-fm)/(2*delta);
end;

begin
  writeln('introduceti aproximatia initiala');
  for i:=1 to 3 do
    begin write('x[',i,']= '); readln(x[i]); end;
  sisiter(3,f,df,x,iter); writeln;
  case iter of
  1..99: writeln('nr de iteratii este ',iter);
  100:   begin;
         writeln('metoda nu e convergenta');
         halt; end;
  101:   begin;
         writeln('det=0, sist nedet'); halt; end;
  end;
  writeln;
  for i:=1 to 3 do writeln('x[',i,']= ',x[i]);
  readln;
end.
```

10) <u>Probleme</u>

1) Să se rezolve sistemul :

$$\begin{cases} 0.03x + 3y = 2.5 \\ 2x + 0.05y = 6 \end{cases}$$

prin metoda Gauss , în două variante :

a) asigurând pivotul maxim

b) neasigurând pivotul maxin .

Să se compare rezultatele .

2) Folosind metoda Gauss de rezolvare a sistemelor de ecuații liniare , să se scrie câte un program pentru fiecare din variantele :

a) rezolvă sistemul fără să aleagă pivotul maxim

b) alege pivotul maxim din coloana C(k)

c) alege pivotul maxim din linia L(k)

3) Să se rezolve sistemul :

$$\begin{cases} x = 1 + ye^{x} \\ y^{2} = 2 + xe^{-y} \end{cases}$$

luând aproximația inițială x=0 și y=-1 .

4) Elaborați un program care să rezolve un

sistem de ecuații neliniare prin metoda gradientului .

5) Realizați două proceduri distincte : una care să calculeze soluția unui sistem de ecuații liniare , și cealaltă care să inverseze o matrice , pornind de la procedura sismat .

Ambele proceduri vor trebui să verifice că determinantul este nenul , deci vor avea și parametrul de ieșire det .

CAP IX. STATISTICA DATELOR EXPERIMENTALE

1) Functii de distributie ale valorilor masurate

O măsurătoare experimentală se caracterizează nu numai prin valoarea numerică obținută pentru mărimea măsurată , dar și prin exactitatea acestei valori (adică diferența dintre valoarea măsurată și cea reală , numită și eroare de măsură). In procesul de măsurare a unei mărimi fizice exactitatea rezultatului este afectată de trei factori :

a) precizia aparatului de măsură
b) limita teoretică de precizie permisă
c) metoda de măsură

Datorită faptului că la scară microscopică materia se supune legilor mecanicii cuantice , nu putem calcula decât probabilități de evoluție a sistemului ,cunoașterea fiind limitată de către principiul de incertitudine al lui Heisemberg (nu putem avea o perfectă determinare simultană a tuturor mărimilor fizice ce descriu sistemul). Aceasta este limita teoretică a cunoașterii . In practică însă fluctuațiile la scară microscopică afectează și comportarea aparatului de măsură , care este un sistem macroscopic . Ecuațiile mecanicii cuantice neputând fi rezolvate pentru astfel de sisteme , tot ce putem spune este că rezultatul măsurării mărimii fizice va fi alterat , apărând o eroare introdusă de aparat . Cu cât această eroare este mai mică , cu atât precizia aparatului de măsură este mai mare . Metoda de măsură introduce o eroare sistematică ce poate fi eliminată prin corecții . Fluctuațiile statistice descrise mai sus însă nu pot fi eliminate , ci numai puse în evidență și micșorate prin repetarea măsurătorilor și medierea rezultatelor.

In cadrul unei experiențe se încearcă obținerea "valorii adevărate" a unei mărimi fizice . De fapt valoarea adevărată nu se poate obține niciodată , se poate obține numai o aproximație a ei .

In cazul în care mărimea fizică ia valori discrete , dispersia (împrăștierea) valorilor măsurate y_i va fi dată de probabilitatea $P(y_i)$, pentru fiecare valoare posibilă de obținut y_i . In cazul în care mărimea fizică este continuă vom avea o densitate de probabilitate $P(y)$, probabilitatea ca valoarea măsurată să fie in intervalul $(y,y+dy)$ fiind $P(y)\cdot dy$.

Cu certitudine valoarea măsurată va fi printre cele posibile , deci vom obține probabilitate 1 pentru obținerea oricărui rezultat din cele posibile . Folosind teorema de adunare a probabilităților evenimentelor incompatibile ,

obținem <u>condiția de normare la unitate</u> a funcției de distribuție :

$$\sum_i P(y_i) = 1 \qquad\qquad \int_{-\infty}^{\infty} P(y)\cdot dy = 1$$

De interes practic în analiza datelor sunt următoarele valori :

<u>media</u>

$$\langle y\rangle = \sum_i y_i\cdot P(y_i) \qquad\qquad \langle y\rangle = \int_{-\infty}^{\infty} y\cdot P(y)\cdot dy$$

<u>media patratica</u>

$$\langle y^2\rangle = \sum_i y_i^2\cdot P(y_i) \qquad\qquad \langle y^2\rangle = \int_{-\infty}^{\infty} y^2\cdot P(y)\cdot dy$$

<u>valoarea cea mai probabila</u> se obține rezolvând ecuația :

$$\frac{dP(y)}{dy} = 0 \Rightarrow y_p$$

<u>dispersia (abaterea patratica medie)</u>

$$\sigma^2 = \langle (y-\langle y\rangle)^2\rangle = \sum_i (y_i-\langle y\rangle)^2\cdot P(y_i) =$$

$$= \sum_i y_i^2\cdot P(y_i) + \langle y\rangle^2\cdot\sum_i P(y_i) - 2\langle y\rangle\cdot\sum_i y_i\cdot P(y_i) =$$

$$= \langle y^2\rangle + \langle y\rangle^2 - 2\langle y\rangle^2 = \langle y^2\rangle - \langle y\rangle^2$$

$$\sigma^2 = \langle (y-\langle y\rangle)^2\rangle = \int_{-\infty}^{\infty} (y-\langle y\rangle)^2\cdot P(y)\cdot dy =$$

$$= \int_{-\infty}^{\infty} y^2\cdot P(y) + \langle y\rangle^2\cdot\int_{-\infty}^{\infty} P(y)\cdot dy -$$

$$- 2\langle y\rangle\cdot\int_{-\infty}^{\infty} y\cdot P(y)\cdot dy =$$

$$= \langle y^2\rangle + \langle y\rangle^2 - 2\langle y\rangle^2 = \langle y^2\rangle - \langle y\rangle^2$$

In continuare vom analiza unele din cele mai întâlnite funcții de distribuție .

2) <u>Distributia binomiala</u>

Să considerăm că în urma efectuării experienței putem obține doar două valori , A și B , cu probabilitățile P(A)=p și P(B)=q . Evident vom avea : p + q = 1 . Se pune întrebarea : care este probabilitatea $P_N(n)$ ca , în urma efectuării a N experiențe , să obținem de n ori valoarea A ,

şi de (N-n) ori valoarea B ?

Să vedem mai întâi care este probabilitatea ca primele n rezultate să fie A , iar următoarele N-n să fie B . Comform teoremei de înmulţire a probabilităţilor evenimentelor independente , probabilitatea totală va fi :

$$p \cdot p \cdot \ldots \cdot p \cdot q \cdot q \cdot \ldots \cdot q = p^n \cdot q^{N-n}$$

Insă ordinea în care apar rezultatele nu este importantă , avem combinări de N luate câte n cazuri în care vom avea n rezultate A , probabilitatea pentru fiecare caz fiind tot $p^n \cdot q^{N-n}$. Comform teoremei de adunare a probabilităţilor evenimentelor incompatibile , vom suma peste toate aceste cazuri , obţinând :

$$P_N(n) = C_N^n \cdot p^n \cdot q^{N-n} \qquad (1)$$

unde $C_N^n = \frac{N!}{n!(N-n)!}$

Să verificăm condiţia de normare la unitate :

$$\sum_{n=0}^{N} P_N(n) = \sum_{n=0}^{N} C_N^n \cdot p^n \cdot q^{N-n} = (p+q)^N = 1$$

Am folosit relaţia p+q=1 şi formula binomului lui Newton :

$$\sum_{n=0}^{N} C_N^n \cdot p^n \cdot q^{N-n} = (p+q)^N$$

In continuare vom calcula media şi dispersia acestei distribuţii .

$$\langle n \rangle = \sum_{n=0}^{N} n \cdot \frac{N!}{n!(N-n)!} \cdot p^n \cdot q^{N-n} =$$

$$= N \cdot p \cdot \sum_{n=1}^{N} \frac{(N-1)!}{(n-1)!(N-n)!} \cdot p^{n-1} \cdot q^{N-n} =$$

renotare $n-1 \Rightarrow n$

$$= N \cdot p \cdot \sum_{n=0}^{N-1} \frac{(N-1)!}{n!(N-1-n)!} \cdot p^n \cdot q^{N-1-n} = N \cdot p \cdot (p+q)^{N-1} =$$

$$= N \cdot p$$

$$\langle n^2 \rangle = \sum_{n=0}^{N} n^2 \cdot \frac{N!}{n!(N-n)!} \cdot p^n \cdot q^{N-n} =$$

$$= \sum_{n=0}^{N} [n(n-1)+n] \cdot \frac{N!}{n!(N-n)!} \cdot p^n \cdot q^{N-n} =$$

$$= N(N-1)\cdot p^2\cdot \sum_{n=2}^{N} \frac{(N-2)!}{(n-2)!(N-n)!}\cdot p^{n-2}\cdot q^{N-n} + <n> =$$

renotare $n-2 \Rightarrow n$

$$= N(N-1)\cdot p^2\cdot \sum_{n=0}^{N-2} \frac{(N-2)!}{n!(N-2-n)!}\cdot p^{n}\cdot q^{N-2-n} + <n> =$$

$$= N(N-1)\cdot p^2\cdot (p+q)^{N-2} + N\cdot p = N(N-1)\cdot p^2 + N\cdot p$$

$$\sigma^2 = <n^2> - <n>^2 = N(N-1)\cdot p^2 + N\cdot p - N^2\cdot p^2 =$$

$$= N\cdot p(1-p) = N\cdot p\cdot q$$

3) <u>Distributia</u> <u>Poisson</u>

Distribuția Poisson se obține ca o aproximație a distribuției binomiale , pentru cazul când :

a) $p<<1$

b) $n<<N$

In aceste condiții avem următoarele două aproximații :

$N! \approx N^n\cdot(N-n)!$

$q^{N-n} = (1-p)^{N-n} \approx (1-p)^N \approx [\exp(-p)]^N = e^{-pN}$

Și deci formula devine :

$$P(n) = \frac{(pN)^n}{n!}\cdot e^{-pN}$$

Ținând cont de ipotezele de lucru , impunem $N\Rightarrow\infty$ și $p\Rightarrow 0$ astfel încât $pN = m$. Distribuția devine :.

$$P(n) = \frac{m^n}{n!}\cdot e^{-m} \qquad (2)$$

Să verificăm condiția de normare la unitate :

$$\sum_{n=0}^{\infty} P(n) = e^{-m}\cdot \sum_{n=0}^{\infty} \frac{m^n}{n!} = e^{-m}\cdot e^{m} = 1$$

Am folosit formula dezvoltării în serie a exponențialei :

$$\sum_{n=0}^{\infty} \frac{m^n}{n!} = e^m$$

In continuare vom calcula media și dispersia acestei distribuții .

$$<n> = \sum_{n=0}^{\infty} n\cdot\frac{m^n}{n!}\cdot e^{-m} = m\cdot e^{-m}\cdot \sum_{n=1}^{\infty} \frac{m^{n-1}}{(n-1)!} =$$

$$= m\cdot e^{-m}\cdot e^{m} = m$$

$$<n^2> = \sum_{n=0}^{\infty} n^2\cdot\frac{m^n}{n!}\cdot e^{-m} = e^{-m}\cdot \sum_{n=0}^{\infty} [n(n-1)+n]\cdot\frac{m^n}{n!} =$$

$$= e^{-m}\cdot m^2\cdot \sum_{n=2}^{\infty} \frac{m^{n-2}}{(n-2)!} + <n> = e^{-m}\cdot m^2\cdot e^m + m = m^2 + m$$

$$\sigma^2 = <n^2> - <n>^2 = m^2 + m - m^2 = m$$

Observăm că aceste valori coincid cu cele date de distribuția binomială , în limita $N\Rightarrow\infty$, $p\Rightarrow 0$, astfel încât $N\cdot p=m=$constant .

Distribuția Poisson așadar se scrie :

$$P(n) = \frac{<n>^n}{n!}\cdot e^{-<n>} \quad \text{având } \sigma^2=<n>$$

4) Distributia Gauss (distributia normala)

Distribuția Gauss se obține ca o aproximație a distribuției binomiale , pentru cazul când :

a) $1<<N$

b) $1<<n$

In această condiție în formulele mediilor putem trece de la sumare la integrare , distribuția discretă trecând într-una continuă .

$$P(n) = \frac{N!}{n!(N-n)!}\cdot p^n\cdot q^{N-n}$$

$$P(n+1) = \frac{N!}{(n+1)!(N-n-1)!}\cdot p^{n+1}\cdot q^{N-n-1}$$

$$\frac{dP(n)}{dn} \approx \frac{P(n+1) - P(n)}{(n+1) - n} = P(n)\cdot(\frac{(N-n)p}{(n+1)q} -1) =$$

$$= P(n)\cdot\frac{Np-n-q}{nq+q} \approx P(n)\cdot\frac{Np-n}{nq}$$

Din această ecuație observăm că $P(n)$ își atinge maximul în $n=Np$.

Deoarece n și N sunt foarte mari, funcția $P(n)$ are o variație foarte lentă . Funcția $\ln(P(n))$, care își atinge și ea extremul în $n=Np$, are o variație și mai lentă (prima derivată e foarte mică) . Putem presupune că în jurul punctului de extrem a doua derivată a logaritmului e constantă , și anume egală cu valoarea ei în punctul de extrem (curbura funcției e aproximativ aceeași) .

$$\frac{dP(n)}{dn} = P(n)\cdot\frac{Np-n}{nq} \Rightarrow \frac{d\ \ln(P(n))}{dn} = \frac{Np-n}{nq} \Rightarrow$$

$$\Rightarrow \frac{d^2\ \ln(P(n))}{dn^2} = \frac{-1}{nq} - \frac{Np-n}{n^2q}$$

$$\frac{d^2\ \ln(P(n))}{dn^2} \approx \frac{d^2\ \ln(P(n=Np))}{dn^2} = \frac{-1}{Npq} \Rightarrow$$

$$\Rightarrow \frac{d^2\ \ln(P(n))}{dn^2} = \frac{-1}{Npq}$$

Integrăm această ecuație :

$$\frac{d\ \ln(P(n))}{dn} = \frac{-n}{Npq} + A$$

unde A este constanta de integerare .

$$\frac{d\ \ln(P(n=Np))}{dn} = 0 \Rightarrow A=\frac{1}{q} \Rightarrow \frac{d\ \ln(P(n))}{dn} = \frac{Np-n}{Npq}$$

Integrăm din nou :

$$\ln(P(n)) = \frac{-(n-Np)^2}{2Npq} + B$$

unde B este constanta de integrare .

$$P(n) = C\cdot\exp\left(\frac{-(n-Np)^2}{2Npq}\right) \qquad \text{cu } C=\exp(B)$$

Constanta C o obținem din condiția de normare:

$$\int_{-\infty}^{\infty} P(n)\cdot dn = 1 \Rightarrow C^{-1} = \int_{-\infty}^{\infty} \exp\left(\frac{-(n-Np)^2}{2Npq}\right)\cdot dn =$$

$$= \int_{-\infty}^{\infty} \exp\left(\frac{-x^2}{2Npq}\right)\cdot dx = 2\cdot\sqrt{2Npq}\cdot\int_{0}^{\infty} \exp(-y^2)\cdot dy =$$

$$= 2\cdot\sqrt{2Npq}\cdot\frac{1}{2}\cdot\int_{0}^{\infty} z^{-1/2}\cdot e^{-z}\cdot dz = \sqrt{2\pi Npq}$$

unde transformările de variabilă au fost : $x=n-Np$, $y=x/\sqrt{2Npq}$, $z=y^2$, și am folosit funcția lui Euler $\Gamma(1/2)=\sqrt{\pi}$. Distribuția devine :

$$P(n) = \frac{1}{\sqrt{2\pi Npq}}\cdot\exp\left(\frac{-(n-Np)^2}{2Npq}\right)$$

In continuare vom calcula media și dispersia acestei distribuții .

$$\langle n\rangle = \frac{1}{\sqrt{2\pi Npq}}\cdot\int_{-\infty}^{\infty} n\cdot\exp\left(\frac{-(n-Np)^2}{2Npq}\right)\cdot dn =$$

$$= \frac{1}{\sqrt{2\pi Npq}}\cdot\int_{-\infty}^{\infty} (n-Np)\cdot\exp\left(\frac{-(n-Np)^2}{2Npq}\right)\cdot dn +$$

$$+ Np\cdot\frac{1}{\sqrt{2\pi Npq}}\cdot\int_{-\infty}^{\infty} \exp\left(\frac{-(n-Np)^2}{2Npq}\right)\cdot dn = Np$$

Prima integrală dă zero deoarece în urma schimbării de variabilă : $v=n-Np$ obținem integrant impar și domeniu de integrare simetric.

$$\langle n^2\rangle = \frac{1}{\sqrt{2\pi Npq}}\cdot\int_{-\infty}^{\infty} n^2\cdot\exp\left(\frac{-(n-Np)^2}{2Npq}\right)\cdot dn =$$

$$= \frac{1}{\sqrt{2\pi Npq}} \cdot \int_{-\infty}^{\infty} [(n-Np)^2 - (Np)^2 + 2nNp] \cdot$$

$$\cdot \exp(\frac{-(n-Np)^2}{2Npq}) \cdot dn =$$

$$= \frac{1}{\sqrt{2\pi Npq}} \cdot \int_{-\infty}^{\infty} x^2 \cdot \exp(\frac{-x^2}{2Npq}) \cdot dx - (Np)^2 + 2Np \cdot Np =$$

$$= \frac{2 \cdot (2Npq)^{3/2}}{\sqrt{2\pi Npq}} \cdot \int_0^{\infty} y^2 \cdot \exp(-y^2) \cdot dy + (Np)^2 =$$

$$= \frac{4Npq}{\sqrt{\pi}} \cdot \frac{1}{2} \cdot \int_0^{\infty} z^{1/2} \cdot e^{-z} \cdot dz + (Np)^2 = Npq + (Np)^2$$

am folosit funcția lui Euler $\Gamma(3/2) = \frac{\sqrt{\pi}}{2}$.

$\sigma^2 = \langle n^2 \rangle - \langle n \rangle^2 = Npq + (Np)^2 - (Np)^2 = Npq$

Observăm că aceste valori coincid cu cele date de distribuția binomială .

Distribuția Gauss așadar se scrie :

$$P(n) = \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot \exp(\frac{-(n-\langle n \rangle)^2}{2\sigma^2}) \qquad (3)$$

<u>Observatie</u> <u>importanta</u>

Pentru cazul în care :

a) $p \ll 1$

b) $1 \ll n \ll N$

putem avea ambele aproximații ale distribuției binomiale , și distribuția Poisson este echivalentă cu distribuția Gauss , având desigur amândouă aceeași valoare medie $\langle n \rangle = Np$, și aceeași dispersie $\sigma^2 = Np$. ($p \ll 1 \Rightarrow q = 1-p \approx 1$ în dispersia distribuției Gauss)

5) Distributia erorilor de masura

Această distribuție a fost descoperită de Gauss , în timp ce studia și fundamenta teoria erorilor . Pentru a o deduce vom porni de la următoarele postulate :

1) Rezultatele măsurătorilor sunt afectate în mod inerent de erori întâmplătoare .

2) Abaterea rezultatului x de la valoarea adevărată a mărimii măsurate este cauzată de un număr foarte mare N de factori aleatori , fiecare din aceștia provocând o eroare elementară ε foarte mică .

3) Cauzele care provoacă apariția erorilor întâmplătoare sunt independente între ele .

4) Probabilitatea de apariție a erorilor

pozitive $(+\varepsilon)$ este egală cu probabilitatea de apariție a erorilor negative $(-\varepsilon)$.

Deoarece o eroare poate să fie sau pozitivă (cu probabilitatea p) sau negativă (cu probabilitatea q) , rezultă din 4) că $p=q=1/2$.

Probabilitatea ca din cele N erori elementare n să fie pozitive și (N-n) să fie negative este dată de distribuția binomială :

$$P_N(n) = \frac{N!}{n!(N-n)!}\cdot\left(\frac{1}{2}\right)^n\cdot\left(\frac{1}{2}\right)^{N-n} = \frac{N!}{n!(N-n)!}\cdot\left(\frac{1}{2}\right)^N$$

Eroarea totală va fi :

$\Delta_n = n\varepsilon + (N-n)(-\varepsilon) = (2n-N)\varepsilon$

Probabilitatea ca să avem (n+1) erori elementare pozitive este :

$$P_N(n+1) = \frac{N!}{(n+1)!(N-n-1)!}\cdot\left(\frac{1}{2}\right)^N$$

Eroarea totală va fi :

$\Delta_{n+1} = (n+1)\varepsilon + (N-n-1)(-\varepsilon) = (2n+2-N)\varepsilon$

Pentru valori ale n și N foarte mari , și ale lui ε foarte mici , trecem de la cazul discret la cazul continuu :

$$\Delta = \frac{\Delta_n + \Delta_{n+1}}{2} = (2n-N+1)\varepsilon$$

este mijlocul intervalului (Δ_n, Δ_{n+1})

$$\frac{d\Delta}{dn} = \frac{\Delta_{n+1} - \Delta_n}{(n+1) - n} = 2\varepsilon$$

este lungimea intervalului (Δ_n, Δ_{n+1})

Aproximăm că P(n) are o variație liniară în intervalul (Δ_n, Δ_{n+1}) .

$$P(\Delta) = \frac{P_N(n) + P_N(n+1)}{2} = \frac{P_N(n)}{2}\cdot\left(1 + \frac{N-n}{n+1}\right) =$$

$$= P_N(n)\cdot\frac{N+1}{2(n+1)}$$

este probabilitatea ca Δ să fie la mijlocul intervalului (Δ_n, Δ_{n+1})

$$\frac{dP}{dn} = \frac{P_N(n+1) - P_N(n)}{(n+1) - n} = P_N(n)\cdot\left(\frac{N-n}{n+1} - 1\right) =$$

$$= P_N(n)\cdot\frac{N-2n-1}{n+1}$$

Trecem de la variabila n la variabila Δ (eroarea) :

$$\frac{dP}{d\Delta} = \frac{dP}{dn}\cdot\frac{dn}{d\Delta} = P_N(n)\cdot\frac{N-2n-1}{n+1}\cdot\frac{1}{2\varepsilon} =$$

$$= P(\Delta)\cdot\frac{2(n+1)}{N+1}\cdot\frac{-(2n-N+1)}{n+1}\cdot\frac{\varepsilon}{2\varepsilon^2} =$$

$$= P(\Delta)\cdot\frac{-\Delta}{(N+1)\cdot\varepsilon^2} \Rightarrow \frac{dP}{P(\Delta)} = -\frac{\Delta\cdot d\Delta}{(N+1)\cdot\varepsilon^2} = -\frac{\Delta\cdot d\Delta}{\sigma^2}$$

Am considerat limita $n\Rightarrow\infty$, $N\Rightarrow\infty$, $\varepsilon\Rightarrow 0$, astfel încât $(N+1)\cdot\varepsilon^2 = \sigma^2$.

Funcția P reprezintă probabilitatea . Noi căutăm funcția $f(\Delta)$ ce este densitatea de probabilitate . Vom avea :

$P(\Delta) = f(\Delta)\cdot d\Delta$

Diferențiind obținem :

$dP = df(\Delta)\cdot d\Delta$

Prin împărțire obținem :

$$\frac{df(\Delta)}{f(\Delta)} = \frac{dP}{P(\Delta)} \Rightarrow \frac{df(\Delta)}{f(\Delta)} = -\frac{\Delta\cdot d\Delta}{\sigma^2} \Rightarrow$$

$$\Rightarrow d(\ln(f(\Delta))) = -\frac{\Delta\cdot d\Delta}{\sigma^2}$$

Integrând obținem :

$$\ln(f(\Delta)) = -\frac{\Delta^2}{2\cdot\sigma^2} + B$$

unde B este constanta de integrare

$$f(\Delta) = C\cdot \exp(-\frac{\Delta^2}{2\cdot\sigma^2}) \quad \text{cu } C=\exp(B)$$

Constanta C o scoatem din condiția de normare:

$$\int_{-\infty}^{\infty} f(\Delta)\cdot d\Delta = 1 \Rightarrow C = \frac{1}{\sqrt{2\pi}\cdot\sigma}$$

Am obținut deci densitatea de repartiție a erorii aleatoare Δ sub forma unei distribuții Gauss :

$$f(\Delta) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot \exp(-\frac{\Delta^2}{2\cdot\sigma^2}) \qquad (4)$$

Observăm simetria funcției ($f(-\Delta)=f(\Delta)$) și faptul că își atinge maximul în $\Delta=0$ (aceasta este valoarea cea mai probabilă și în același timp și valoarea medie a erorii) .

Probabilitățile ca erorile să fie cuprinse în următoarele intervale sunt :

$$P(-\sigma\le\Delta\le\sigma) = \int_{-\sigma}^{\sigma} f(\Delta)\cdot d\Delta = 0.6826$$

$$P(-2\sigma\le\Delta\le2\sigma) = \int_{-2\sigma}^{2\sigma} f(\Delta)\cdot d\Delta = 0.9546$$

$$P(-3\sigma\le\Delta\le3\sigma) = \int_{-3\sigma}^{3\sigma} f(\Delta)\cdot d\Delta = 0.9974$$

$$P(-4\sigma\le\Delta\le4\sigma) = \int_{-4\sigma}^{4\sigma} f(\Delta)\cdot d\Delta = 0.9994$$

Să notăm cu $\underline{x}$ valoarea adevărată a mărimii măsurate . Probabilitatea de a avea eroarea Δ

este identică cu cea de a măsura valoarea x , cu $x = \bar{x} + \Delta$. Inlocuim în distribuția erorilor și obținem :

$$f(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp(-\frac{(x-\bar{x})^2}{2\cdot\sigma^2}) \qquad (5)$$

Am obținut deci că rezultatul valorii măsurate este distribuit după o funcție Gauss în jurul valorii adevărate .

<u>Observatie importanta</u>

Fie variabila x distribuită după o funcție Gauss , ca mai sus , și o funcție $y=y(x)$. Să cercetăm care este distribuția lui y .

In cazul în care y este injectivă , intervalul $[x,x+dx]$ se va duce în intervalul $[y,y+dy]$, cu $y=y(x)$ și $dy=y'(x)\cdot dx$. In cazul în care funcția nu este injectivă , mai multe intervale $[x,x+dx]$ se vor duce în $[y,y+dy]$, și distribuția lui y devine foarte complicată . Noi vom lucra în cazul funcției injective , astfel că putem scrie :

$x=y^{-1}(y)$

$P(x\in[x,x+dx]) = f(x)\cdot dx$

$P(y\in[y,y+dy]) = g(y)\cdot dy$

$P(x\in[x,x+dx]) = P(y\in[y,y+dy])$ datorită injectivității

$$\Rightarrow f(x)\cdot dx = g(y)\cdot dy \Rightarrow g(y) = \frac{f(x)}{y'(x)} \Rightarrow$$

$$\Rightarrow g(y) = \frac{f\circ y^{-1}(y)}{y'\circ y^{-1}(y)}$$

In cazul dependenței liniare $y=ax+b$ vom avea :

$y'(x) = a = \text{constant}$

$y^{-1}(y) = \frac{y-b}{a}$

$$g(y) = \frac{1}{\sqrt{2\pi}\cdot\sigma a}\cdot\exp(-\frac{(\frac{y-b}{a} - \bar{x})^2}{2\cdot\sigma^2}) =$$

$$= \frac{1}{\sqrt{2\pi}\cdot\bar{\sigma}}\cdot\exp(-\frac{(y-\bar{y})^2}{2\cdot\bar{\sigma}^2})$$

și deci obținem tot o distribuție Gauss , cu $\bar{y} = a\bar{x}+b$ și $\bar{\sigma} = \sigma a$.

In cazul în care y nu depinde liniar de x , repartiția lui nu va fi dată de o funcție Gauss .

6) <u>Legea de propagare a erorilor</u>

Fie $x_1,x_2,\ldots,x_n$ valorile măsurate , $\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_n$ valorile adevărate , și $F(x_1,x_2,\ldots,x_n)$ o funcție .

Valorile $x_1,x_2,\ldots,x_n$ se vor supune unei distribuții Gauss .

Notăm $x=(x_1,x_2,\ldots,x_n)$ și $\bar{x}=(\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_n)$

$P(x) = P_1(x_1)\cdot P_2(x_2)\cdot\ldots\cdot P_n(x_n)$ densitatea de probabilitate

$dx = dx_1\cdot dx_2\cdot\ldots\cdot dx_n$

Descompunem F(x) în serie Taylor în jurul lui $\bar{x}$. In cazul în care putem neglija termenii de grad mai mare decât unu , comform observației importante precedente , distribuția variabilei F(x) este o distribuție normală (Gauss) , și îi putem determina valoarea medie și abate-rea standard .

$$F(x) \approx F(\bar{x}) + \sum_{i=1}^{n} (x_i-\bar{x}_i)\cdot\frac{\partial F(\bar{x})}{\partial x_i} \qquad (6)$$

$$<F(x)> = \int_{(x)} F(x)\cdot P(x)\cdot dx =$$

$$= \int_{(x)} [\ F(\bar{x}) + \sum_{i=1}^{n} (x_i-\bar{x}_i)\cdot\frac{\partial F(\bar{x})}{\partial x_i}\ ]\cdot P(x)\cdot dx =$$

$$= F(\bar{x})\cdot \int_{(x_1)}\cdots\int_{(x_n)} P_1(x_1)\cdot\ldots\cdot P_n(x_n)\cdot dx_1\cdot\ldots\cdot dx_n +$$

$$+ \sum_{i=1}^{n}\frac{\partial F(\bar{x})}{\partial x_i}\cdot\int_{(x_1)}\cdots\int_{(x_n)} (x_i-\bar{x}_i)\cdot$$

$$\cdot P_1(x_1)\cdot\ldots\cdot P_n(x_n)\cdot dx_1\cdot\ldots\cdot dx_n = F(\bar{x})$$

deoarece

$$\int_{(x_i)} P_i(x_i)\cdot dx_i = 1$$

și

$$\int_{(x_i)} (x_i-\bar{x}_i)\cdot P_i(x_i)\cdot dx_i = \bar{x}_i - \bar{x}_i = 0$$

$$\sigma^2 = <(F(x)-F(\bar{x}))^2> =$$

$$= \int_{(x_1)}\cdots\int_{(x_n)} (\sum_{i=1}^{n}(x_i-\bar{x}_i)\cdot\frac{F(\bar{x})}{\partial x_i})^2\cdot$$

$$\cdot P_1(x_1)\cdot\ldots\cdot P_n(x_n)\cdot dx_1\cdot\ldots\cdot dx_n =$$

$$= \sum_{i,j=1}^{n}\frac{\partial F(\bar{x})}{\partial x_i}\cdot\frac{\partial F(\bar{x})}{\partial x_j}\cdot\int_{(x_1)}\cdot\int_{(x_n)} (x_i-\bar{x}_i)\cdot(x_j-\bar{x}_j)\cdot$$

$$\cdot P_1(x_1)\cdot\ldots\cdot P_n(x_n)\cdot dx_1\cdot\ldots\cdot dx_n =$$

$$= \sum_{i=1}^{n} \left(\frac{\partial F(\bar{x})}{\partial x_i}\right)^2 \cdot \int_{(x_i)} (x_i-\bar{x}_i)^2 \cdot P_i(x_i) \cdot dx_i$$

deoarece termenii cu $i \neq j$ sunt nuli . Observăm că integralele rămase sunt tocmai erorile pătratice medii ale variabilelor x_i , $\sigma^2(x_i)$. Deoarece am lucrat în aproximația de ordinul I al dezvoltării funcției $F(x)$ în jurul lui $F(\bar{x})$, avem implicit :

$$\frac{\partial F(\bar{x})}{\partial x_i} \approx \frac{\partial F(x)}{\partial x_i}$$ (prin derivarea în funcție de x_i a ecuației (6) a dezvoltării în serie)

Am obținut astfel legea de propagare a erorilor :

$$\sigma^2(F(x)) = \sum_{i=1}^{n} \left(\frac{\partial F(x)}{\partial x_i}\right)^2 \cdot \sigma^2(x_i) \qquad (7)$$

7) Distributia dezintegrarilor radioactive

Pornim de la ipoteza că un nucleu radioactiv are probabilitatea $\lambda \cdot dt$ să se dezintegreze în timpul dt , probabilitate ce nu depinde de celelalte nuclee , sau de istoria respectivului nucleu (de timp).

Fie $W_0(t)$ probabilitatea ca nucleul să nu se dezintegreze în intervalul de timp $[0,t]$ în care se face măsurătoarea .

Probabilitatea ca nucleul să se dezintegreze în intervalul $[t,t+dt]$ este $W_0(t) \cdot \lambda \cdot dt$ (înmulțim probabilități ale unor evenimente independente : nedezintegrare în $[0,t]$ și dezintegrare în $[t,t+dt]$).

Probabilitatea ca nucleul să nu se dezintegreze în $[t,t+dt]$ este $1-W_0(t) \cdot \lambda \cdot dt$ (avem evenimentul complementar precedentului : nucleul ori se dezintegrează , ori nu se dezintegrează în $[t,t+dt]$).

Probabilitatea ca nucleul să nu se dezintegreze în $[0,t+dt]$ este $W_0(t) \cdot [1-W_0(t) \cdot \lambda \cdot dt]$ (înmulțim probabilități ale unor evenimente in-dependente : nedezintegrare în $[0,t]$ și nedezintegrare în $[t,t+dt]$).

Această probabilitate este tocmai $W_0(t+dt)$.

$$W_0(t+dt) = W_0(t) \cdot [1-W_0(t) \cdot \lambda \cdot dt]$$

$$\frac{dW_0(t)}{dt} = \frac{W_0(t+dt)-W_0(t)}{dt} = -W_0(t) \cdot \lambda \Rightarrow$$

$$\Rightarrow \frac{dW_0(t)}{W_0(t)} = -\lambda \cdot dt \Rightarrow$$

$$\Rightarrow \ln(W_0(t)) = -\lambda \cdot t + C$$

unde C este constanta de integrare

La momentul inițial nucleul era nedezintegrat:

$W_0(t=0) = 1 \Rightarrow \ln(W_0(t=0)) = 0 \Rightarrow C=0 \Rightarrow$

$\Rightarrow W_0(t) = \exp(-\lambda t)$

In cazul existenţei mai multor nuclee N , fiecare dintre acestea are , în timpul t în care se efectuează măsurătoarea , probabilitatea de a se dezintegra $p=(1-e^{-\lambda t})$ şi de a nu se dezintegra $q=e^{-\lambda t}$ (cele două evenimente sunt complementare).

Probabilitatea ca n nuclee să se dezintegreze , iar restul (N-n) să nu se dezintegreze , este dată de distribuţia binomială :

$$P_N(n) = \frac{N!}{n!(N-n)!}\cdot(1-e^{-\lambda t})^n\cdot(e^{-\lambda t})^{N-n}$$

In cazul real avem :

a) n<<N (numărul total de nuclee N e foarte mare , $N\to\infty$)

b) $\lambda t<<1$ (timpul de măsură e mic faţă de timpul de înjumătăţire)

şi putem face aproximaţiile :

$$N! \approx N^n\cdot(N-n)!$$

$$(1-e^{-\lambda t})^n = \left(\frac{e^{\lambda t}-1}{e^{\lambda t}}\right)^n \approx (\lambda t)^n$$ prin dezvoltarea exponenţialei în serie şi neglijarea lui (λt) faţă de unitate

$$(e^{-\lambda t})^{N-n} = e^{-\lambda t(N-n)} \approx e^{-\lambda Nt}$$

Distribuţia devine :

$$P(n) = \frac{N^n}{n!}\cdot(\lambda t)^n\cdot e^{-\lambda Nt} = \frac{(\lambda Nt)^n}{n!}\cdot e^{-\lambda Nt} \qquad (8)$$

ceea ce nu este altceva decât o distribuţie Poisson .

$\langle n\rangle = \lambda Nt$ media

$\sigma^2 = \lambda Nt$ eroarea pătratică medie

In conformitate cu prima observaţie importantă , în cazurile în care $\langle n\rangle\geq 50$, putem aproxima distribuţia numărului de nuclee dezintegrate cu o distribuţie Gauss , având valoarea medie $\langle n\rangle$ şi abaterea standard $\sigma = \sqrt{\langle n\rangle}$. In cazul în care numărul de pulsuri furnizate de detector este n_m , cu $n_m\geq 50$, vom avea , $\langle n\rangle \approx n_m$ şi $\sigma \approx \sqrt{n_m}$.

8) Programul PASCAL

In continuare prezentăm un program ce , având ca date de intrare pe N şi pe p , desenează distribuţia binomială , apoi pe cea Poisson cu m=Np , şi pe cea Gauss cu m=Np şi σ^2=Npq . Utilizându-l se pot constata practic echivalenţele între distribuţii în cazurile particulare ale parametrilor distribuţiei binomiale . Este dat şi unit-ul cu funcţiile matematice folosite . Datorită valorii foarte mari a factorialului , s-au utilizat variabile de tipul extended . Principalul beneficiu al aproximării distribuţiei binomiale şi a celei Poisson cu distribuţii Gauss este tocmai

renunțarea la aceste factoriale , greu de calculat .

```
{$F+,N+,E+}
PROGRAM DISTRIB;
uses graph,crt,graphics,matem7;
var N,i:integer; Ns:string;
    p,d,prob,m,s,x:extended;

FUNCTION PGAUSS(x:real):real;
begin
PGauss:=exp(-sqr(x-m)/(2*s))/sqrt(2*pi*s);
end;

begin
write('1<=N<=100   N = ');readln(N);
write('0<=p<=1     p = ');readln(p);
if (N<1) or (N>100) or (p<0) or (p>1) then halt;
histo:=1; str(N,Ns);
init_graph_mode;
init_window(0,maxx,0,maxy-30,0,N,0,1);
outtextxy(0,maxy-20,concat('Xmin=0 , Xmax=',Ns,
                             ' , Ymin=0 , Ymax=1'));
outtextxy(0,maxy-10,'distributiile :');
setcolor(blue);
outtextxy(200,maxy-10,'binomiala');
for i:=0 to N do begin
  prob:=comb(N,i)*putere(p,i)*putere(1-p,N-i);
  punct_histo(i,prob); end;
setcolor(red);
outtextxy(300,maxy-10,'Poisson'); m:=N*p;
for i:=0 to N do begin
   prob:=exp(-m)*putere(m,i)/fact(i);
   punct_histo(i,prob); end;
setcolor(yellow);
outtextxy(400,maxy-10,'Gauss'); s:=N*p*(1-p);
graph_functie(PGauss,yellow);
repeat until readkey='q'; closegraph;
end.


{$N+,E+}
UNIT MATEM7;

interface

function fact(n:integer):extended;
function comb(n,i:integer):extended;
function putere(baza,exponent:extended):extended;

implementation

FUNCTION FACT;
var d:extended; j:integer;
begin
d:=1; for j:=1 to n do d:=d*j; fact:=d;
end;
```

```
FUNCTION COMB;
var d:extended; j:integer;
begin
d:=1; for j:=1 to i do d:=d*(n-i+j)/j; comb:=d;
end;

FUNCTION PUTERE;
begin
putere:=exp(exponent*ln(baza));
end;

end.
```

9) Problema

Fie $y(x_1,x_2)=x_1 \cdot x_2$, iar $\sigma x_1=0.1$ și $\sigma x_2=0.2$. In punctul $x_1=2$ și $x_2=3$ să se determine y , σy , $\frac{\sigma y}{y}$.

Rezolvare

$y = 2 \cdot 3 = 6$

$\frac{dy}{dx_1} = x_2 = 3$ $\qquad$ $\frac{dy}{dx_2} = x_1 = 2$

Folosim legea de propagare a erorilor :

$$\sigma y = \sqrt{(\frac{dy}{dx_1})^2 \cdot \sigma x_1^2 + (\frac{dy}{dx_2})^2 \cdot \sigma x_2^2} = 0.5$$

$\frac{\sigma y}{y} = 0.083$

Rezultatul se scrie sub forma : $y = 6 \pm 0.5$

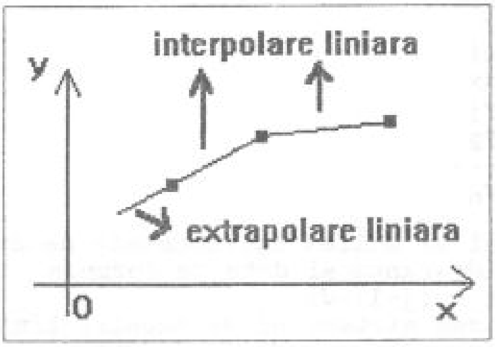
CAP X. INTERPOLAREA EXACTA PRIN POLINOM

1) Interpolarea

Se presupune că se cunoaşte un set de puncte (x_i,y_i) , cu i=1..n , şi se caută o funcţie P(x) astfel încât toate punctele să fie pe graficul acestei funcţii : $y_i = P(x_i)$ pentru orice i=1..n .

Procesul de evaluare a valorii P(x) , cu x dat , fără să cunoaştem funcţia P , numai cu ajutorul valorilor (x_i,y_i) , se numeşte interpolare .

Interpolarea se poate face în mai multe moduri . Cel mai simplu mod este interpolarea liniară , care se foloseşte , de exemplu , în utilizarea tabelelor de logaritmi . Interpolarea liniară constă în aproximarea funcţiei cu segmente de dreaptă care au capetele în două puncte alăturate , aşa cum se vede în figura de mai jos :



Cea mai exactă metodă de interpolare este căutarea funcţiei sub forma unui polinom de grad n-1 , unde n este numărul punctelor date .

$$P(x) = p_1 \cdot x^{n-1} + p_2 \cdot x^{n-2} + .. + p_{n-1} \cdot x + p_n \quad (1)$$

Coeficienţii polinomului sunt scoşi din sistemul de n ecuaţii liniare :

$$p_1 \cdot x_1^{n-1} + p_2 \cdot x_1^{n-2} + .. + p_{n-1} \cdot x_1 + p_n = y_1 \quad (2)$$
$$p_1 \cdot x_2^{n-1} + p_2 \cdot x_2^{n-2} + .. + p_{n-1} \cdot x_2 + p_n = y_2$$
......
$$p_1 \cdot x_n^{n-1} + p_2 \cdot x_n^{n-2} + .. + p_{n-1} \cdot x_n + p_n = y_n$$

Există şi metode de determinare a polinomului P(x) fără a recurge la rezolvarea sistemelor liniare . Astfel polinomul Lagrange de interpolare :

$$L_{n-1}(x) = \sum_{i=1}^{n} c_i(x) \cdot y_i \quad (3)$$

unde

$c_i(x) =$

$$= \frac{(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} \quad (4)$$

trece prin punctele (x_i,y_i) , cu i=1..n , și are gradul n-1 .

Adesea în practică este nevoie să se interpoleze punctele experimentale (x_i,y_i) , cu i=1..n , cu un polinom (sau cu o altă funcție) de grad mai mic decât n-1 . In această situație se folosește metoda celor mai mici pătrate , descrisă în alt capitol .

2) Programul PASCAL

In continuare este dat un program ce citește punctele prin care se fitează cu un polinom , obține coeficienții polinomului prin rezolvarea sistemului de ecuații liniare (2) , și reprezintă grafic punctele și polinomul fitat .

Coordonatele (x_i,y_i) ale punctelor sunt luate dintr-un fișier . Ele pot fi scrise în fișier sub două formate :

XY	Y
x_1	xi
y_1	dx
x_2	y_1
y_2	y_2
...	...
x_n	y_n
y_n	

In cazul al doilea coordonatele pe axa Ox fiind echidistante și date de formula :

x_j = xi + (j-1)·dx

Rezolvarea sistemului de ecuații liniare se face cu ajutorul procedurii sismat din unit-ul sisec , iar reprezentarea grafică se face cu ajutorul procedurilor din unit-ul graphics .

```
{$F+}
PROGRAM interpolare;
uses sisec,graphics,graph,crt;
var a,c:mp; x,y,p:mc; n,i,j:integer; det:real;
    f:text; nume,ordine:string; xi,dx:real;
    x1,x2,y1,y2,d:real;

FUNCTION PUTERE(baza:real;exponent:integer):real;
var i:integer; s:real;
begin
s:=1;
if exponent<0
then for i:=-1 downto exponent do s:=s/baza;
if exponent>0
then for i:=1 to exponent do s:=s*baza;
putere:=s;
end;
```

```
FUNCTION polinom(x:real):real;
var i:integer; s:real;
begin
s:=0;
for i:=1 to n do s:=s+p[i]*putere(x,n-i);
polinom:=s;
end;

begin
writeln('Interpolare exacta prin polinom');
{ CITIREA PUNCTELOR PRIN CARE DUCEM POLIN FITAT }
write('Numele fisierului de date : ');
readln(nume);
assign(f,nume); reset(f); n:=0;
readln(f,ordine);
if (ordine<>'Y') and (ordine<>'XY') then halt;
if ordine='Y'
then begin readln(f,xi); readln(f,dx); end;
while not eof(f) do begin
   n:=n+1;
   if ordine='Y' then x[n]:=xi+(n-1)*dx;
   if ordine='XY' then readln(f,x[n]);
   readln(f,y[n]); end;
close(f);
if (n<1) or (n>10) then halt;
{ REZOLVAREA SIST DE EC PT COEF POLINOMULUI }
for i:=1 to n do
{ pt toate punctele , liniile sist de ec }
for j:=1 to n do
{ pt toti coef polinom , coloanele sist de ec }
   a[i,j]:=putere(x[i],n-j);
sismat(n,a,y,det,p,c);
{ REPREZ GRAFICA A PUNCTELOR SI A POLINOMULUI }
x1:=x[1]; x2:=x[1]; y1:=y[1]; y2:=y[1];
for i:=2 to n do begin
   if x[i]<x1 then x1:=x[i];
   if x[i]>x2 then x2:=x[i];
   if y[i]<y1 then y1:=y[i];
   if y[i]>y2 then y2:=y[i]; end;
d:=x2-x1; x1:=x1-d/10; x2:=x2+d/10;
d:=y2-y1; y1:=y1-d/10; y2:=y2+d/10;
init_graph_mode;
init_window(0,maxx,0,maxy,x1,x2,y1,y2);
for i:=1 to n do star(x[i],y[i]);
graph_functie(polinom,red);
repeat until readkey='q';
closegraph;
end.
```

3) <u>Problema</u>

Alcătuiți un program care , pornind de la un set de puncte , să construiască polinomul Lagrange de interpolare , și să reprezinte grafic punctele și polinomul .

CAP XI. INTERPOLAREA PRIN METODA CELOR MAI MICI PATRATE

1) Testul χ^2

Să presupunem că s-a efectuat măsurarea unui spectru în n puncte : $x_1,x_2,...,x_n$, obținându-se valorile experimentale $y_1,y_2,...,y_n$, care în general diferă de valorile reale : $\bar{y}_1,\bar{y}_2,...,\bar{y}_n$. Dacă presupunem că densitetea de probabilitate $P(y_i)$ ca să se obțină valoarea y_i , în punctul x_i , este de tip Gauss , atunci :

$$P(y_i) = \frac{1}{\sqrt{2\pi}\cdot\sigma_i}\cdot\exp\left(-\frac{(y_i-\bar{y}_i)^2}{2\cdot\sigma_i^2}\right) \qquad (1)$$

In acest caz densitatea de probabilitate $P(y_1,y_2,...,y_n)$ ca pentru setul de valori $x_1,x_2,...,x_n$ să obținem rezultatele $y_1,y_2,...,y_n$ este :

$$P(y_1,y_2,...,y_n) = P(y_1)\cdot P(y_2)\cdot ...\cdot P(y_n) =$$

$$= \frac{1}{\sigma_1\cdot\sigma_2\cdot ...\cdot\sigma_n\cdot(2\pi)^n}\cdot\exp\left(-\frac{1}{2}\sum_{i=1}^{n}\frac{(y_i-\bar{y}_i)^2}{\sigma_i^2}\right) \qquad (2)$$

Observăm că probabilitatea maximă de obținere a valorilor $\{y_i\}$ se realizează atunci când

expresia : $\chi^2 = \sum_{i=1}^{n}\frac{(y_i-\bar{y}_i)^2}{\sigma_i^2}$ este minimă .

In cazul în care legătura dintre x_i și $\bar{y}_i$ este realizată de o funcție analitică cu m parametrii:

$\bar{y}_i = f(x_i,\bar{p}_1,\bar{p}_2,...,\bar{p}_m) = f(x_i,\bar{p})$

$\bar{p} = (p_1,p_2,...,p_m)$

trasăm graficul $y=f(x,\bar{p})$ și plasăm și punctele (x_i,y_i) . Comform observațiilor de la distribuția Gauss , din caracterul statistic al măsurătorilor deducem că doar aproximativ 32% din puncte se vor afla la o distanță de grafic mai mare decât o abatere standard . Dacă abaterile σ_i sunt cunoscute , ele trebuiesc trecute pe grafic sub forma unor segmente verticale de lungime $2\cdot\sigma_i$, cu mijlocul în punctul (x_i,y_i) .

De cele mai multe ori valorile $\bar{y}_i$ nu sunt cunoscute , dacă ar fi fost cunoscute , experiența nu ar mai fi avut rost . In schimb există teorii care prezic o anumită formă analitică pentru funcția $f(x)$, depinzând de anumiți parametrii $\bar{p} = (p_1,p_2,...,p_m)$, ce urmează să fie determinați (cu aproximație) în urma prelucrării datelor experimentale .

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i-\bar{y}_i)^2}{\sigma_i^2} = \sum_{i=1}^{n} \frac{(y_i-f(x_i,\bar{p}))^2}{\sigma_i^2} =$$

$$= \chi^2(\bar{p}) = \chi^2(\bar{p}_1,\bar{p}_2,\ldots,\bar{p}_m)$$

Metoda celor mai mici pătrate impune ca setul de valori $\{y_i\}$ să fie obținut cu probabilitate maximă , adică impune ca funcția χ^2 să fie minimă în raport cu parametrii $p_1,p_2,\ldots,p_m$. Va rezulta sistemul de ecuații :

$$\frac{\partial\chi^2(p)}{\partial p_j} = 0 \quad \text{pentru } j=1..m \tag{3}$$

Rezolvând acest sistem vom obține valorile cele mai probabile $\tilde{p}_1,\tilde{p}_2,\ldots,\tilde{p}_m$, și nu pe cele adevărate $\bar{p}_1,\bar{p}_2,\ldots,\bar{p}_m$.

2) Fitarea cu o constanta

Cazul cel mai simplu este când f(x) este o constantă . Fie deci :

$$f(x) = \bar{p}_1 = \bar{a}$$

$$\chi^2(a) = \sum_{i=1}^{n} \frac{(y_i-a)^2}{\sigma_i^2}$$

$$\frac{\partial\chi^2(a)}{\partial a} = \sum_{i=1}^{n} \frac{2\cdot(y_i-a)\cdot(-1)}{\sigma_i^2} = 0 \Rightarrow \tilde{a} = \frac{\sum_{i=1}^{n} \frac{y_i}{\sigma_i^2}}{\sum_{i=1}^{n} \frac{1}{\sigma_i^2}} \tag{4}$$

Am obținut o medie ponderată , în care ponderea punctului i este :

$$Po_i = \frac{1}{\sigma_i^2}$$

In cazul în care ponderile sunt egale , obținem media aritmetică :

$$\tilde{a} = \frac{1}{n}\cdot\sum_{i=1}^{n} y_i$$

3) Fitarea cu o dreapta

In acest caz f(x) este un polinom de gradul I . Fie deci :

$$f(x) = p_1 + p_2\cdot x = a + b\cdot x$$

$$\chi^2(a,b) = \sum_{i=1}^{n} \frac{(y_i-a-b\cdot x_i)^2}{\sigma_i^2}$$

$$\frac{\partial\chi^2(a,b)}{\partial a} = \sum_{i=1}^{n} \frac{2\cdot(y_i-a-b\cdot x_i)\cdot(-1)}{\sigma_i^2} = 0$$

$$\frac{\partial\chi^2(a,b)}{\partial b} = \sum_{i=1}^{n} \frac{2\cdot(y_i-a-b\cdot x_i)\cdot(-x_i)}{\sigma_i^2} = 0$$

Prelucrăm ecuațiile :

$$a\sum_{i=1}^{n}\frac{1}{\sigma_i^2} + b\sum_{i=1}^{n}\frac{x_i}{\sigma_i^2} = \sum_{i=1}^{n}\frac{y_i}{\sigma_i^2}$$

$$a\sum_{i=1}^{n}\frac{x_i}{\sigma_i^2} + b\sum_{i=1}^{n}\frac{x_i^2}{\sigma_i^2} = \sum_{i=1}^{n}\frac{x_i\cdot y_i}{\sigma_i^2}$$

Notăm :

$$a_{1,1} = \sum_{i=1}^{n}\frac{1}{\sigma_i^2} \quad a_{1,2} = a_{2,1} = \sum_{i=1}^{n}\frac{x_i}{\sigma_i^2} \quad a_{2,2} = \sum_{i=1}^{n}\frac{x_i^2}{\sigma_i^2}$$

$$b_1 = \sum_{i=1}^{n}\frac{y_i}{\sigma_i^2} \quad b_2 = \sum_{i=1}^{n}\frac{x_i\cdot y_i}{\sigma_i^2}$$

Rezolvăm sistemul de două ecuații liniare cu două necunoscute :

$a_{1,1}\cdot a + a_{1,2}\cdot b = b_1$

$a_{2,1}\cdot a + a_{2,2}\cdot b = b_2$

Rezultă soluția :

$$a = \frac{b_1\cdot a_{2,2} - b_2\cdot a_{1,2}}{a_{1,1}\cdot a_{2,2} - a_{2,1}\cdot a_{1,2}} \qquad (5)$$

$$b = \frac{a_{1,1}\cdot b_2 - a_{2,1}\cdot b_1}{a_{1,1}\cdot a_{2,2} - a_{2,1}\cdot a_{1,2}}$$

4) Fitarea cu un polinom

In acest caz f(x) este un polinom de gradul m-1 .Fie deci :

$$f(x) = p_1 + p_2\cdot x + p_3\cdot x^2 + \ldots + p_m\cdot x^{m-1} = \sum_{j=1}^{m} p_j\cdot x^{j-1}$$

$$\chi^2(p) = \sum_{i=1}^{n}\frac{\left(y_i - \sum_{j=1}^{m} p_j\cdot x_i^{j-1}\right)^2}{\sigma_i^2}$$

$$\frac{\partial\chi^2(p)}{\partial p_k} = \sum_{i=1}^{n} \frac{2\cdot(y_i - \sum_{j=1}^{m} p_j\cdot x_i^{j-1})\cdot(-x_i^{k-1})}{\sigma_i^2} = 0$$

pentru k=1..m

Prelucrăm ecuațiile , inversând sumele după i și j :

$$\sum_{j=1}^{m}(p_j \sum_{i=1}^{n}\frac{x_i^{j+k-2}}{\sigma_i^2}) = \sum_{i=1}^{n}\frac{y_i\cdot x_i^{k-1}}{\sigma_i^2} \qquad k=1..m$$

Notăm :

$$a_{k,j} = \sum_{i=1}^{n}\frac{x_i^{j+k-2}}{\sigma_i^2} \qquad b_k = \sum_{i=1}^{n}\frac{y_i\cdot x_i^{k-1}}{\sigma_i^2} \qquad j,k=1..m$$

Rezolvăm sistemul de m ecuații liniare cu m necunoscute :

$$\sum_{j=1}^{m} a_{k,j}\cdot p_j = b_k \qquad k=1..m \qquad (\ AP=B \text{ matricial })$$

Rezultă soluția (prin metoda Kramer) :

$$p_j = \frac{\begin{vmatrix} a_{1,1} & \cdots & b_1 & \cdots & a_{1,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & \cdots & b_m & \cdots & a_{m,m} \end{vmatrix}}{\begin{vmatrix} a_{1,1} & \cdots & a_{1,j} & \cdots & a_{1,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & \cdots & a_{m,j} & \cdots & a_{m,m} \end{vmatrix}} = \frac{|D_j|}{|A|} \qquad (6)$$

pentru j=1..m

unde $A = (a_{k,j})_{k,j=1..m}$, $B = (b_k)_{k=1..m}$ și D_j provine din matricea A prin înlocuirea coloanei j cu coloana matricei B .

5) Fitarea cu o functie oarecare

In acest caz f(x) este o funcție oarecare , cu m parametrii .

$$f(x) = f(x,p_1,p_2,\ldots p_m) = f(x,p)$$

$$\chi^2(p) = \sum_{i=1}^{n}\frac{(y_i - f(x_i,p))^2}{\sigma_i^2}$$

$$\frac{\partial\chi^2(p)}{\partial p_k} = \sum_{i=1}^{n}\frac{2\cdot(y_i - f(x_i,p))}{\sigma_i^2}\cdot\frac{\partial f(x_i,p)}{\partial p_k} = 0 \qquad k=1..m$$

Acesta este un sistem de m ecuații neliniare , pe care il vom rezolva prin metoda Gauss-Newton .

Dezvoltăm funcția f(x,p) în serie Taylor în

jurul punctului p^0 al aproximației inițiale, reținând termenii de ordinul I :

$$f(x,p) = f(x,p^0) + \sum_{j=1}^{m} (p_j-p_j^0)\cdot\frac{\partial f(x,p^0)}{\partial p_j} \qquad (*)$$

Prin derivare în funcție de p_k rezultă :

$$\frac{\partial f(x,p)}{\partial p_k} = \frac{\partial f(x,p^0)}{\partial p_k}$$

Sistemul de ecuații devine :

$$\sum_{i=1}^{n}\left[y_i-f(x_i,p^0)-\sum_{j=1}^{m}(p_j-p_j^0)\cdot\frac{\partial f(x_i,p^0)}{\partial p_j}\right]\cdot$$

$$\cdot\frac{\partial f(x_i,p^0)}{\partial p_k}\cdot\frac{1}{\sigma_i^2} = 0 \qquad k=1..m$$

Prelucrăm ecuațiile, inversând sumele după i și j :

$$\sum_{j=1}^{m}\left[(p_j-p^0_j)\sum_{i=1}^{n}\frac{\partial f(x_i,p^0)}{\partial p_j}\cdot\frac{\partial f(x_i,p^0)}{\partial p_k}\cdot\frac{1}{\sigma_i^2}\right] =$$

$$= \sum_{i=1}^{n}(y_i-f(x_i,p^0)\cdot\frac{\partial f(x_i,p^0)}{\partial p_k}\cdot\frac{1}{\sigma_i^2}$$

Notăm : $\Delta p_j = p_j-p_j^0$

$$a_{k,j} = \sum_{i=1}^{n}\frac{\partial f(x_i,p^0)}{\partial p_j}\cdot\frac{\partial f(x_i,p^0)}{\partial p_k}\cdot\frac{1}{\sigma_i^2} \qquad j,k=1..m$$

$$b_k = \sum_{i=1}^{n}(y_i-f(x_i,p^0)\cdot\frac{\partial f(x_i,p^0)}{\partial p_k}\cdot\frac{1}{\sigma_i^2} \qquad k=1..m$$

Avem matricile :

$A = (a_{k,j})_{k,j=1..m}$, $B = (b_k)_{k=1..m}$

Rezolvăm sistemul de m ecuații liniare cu m necunoscute :

$$\sum_{j=1}^{m} a_{k,j}\cdot\Delta p_j = b_k \qquad k=1..m \quad (\ A\Delta=B \text{ matricial })$$

Soluțiile acestui sistem ne furnizează corecțiile la valorile parametrilor din aproximația inițială : $\Delta p_j^1 = \Delta p_j$.

$p_j^1 = p_j^0 + \Delta p_j^1$

Acești parametrii p_j^1 încă nu vor fi egali cu parametrii p_j care să minimalizeze funcția $\chi^2(p)$, deoarece în obținerea sistemului corecțiilor s-au

neglijat termenii de ordin superior lui unu . Totuși , în general , valorile p_j^1 sunt mai apropiate de $\tilde{p}_j$ decât p_j^0 . Pentru a determina și mai exact valorile p_j , rezolvarea sistemului se repetă , înlocuind pe p^0 cu p^1 . Acest proces se repetă până când se atinge numărul maxim de iterații , sau până când raportul dintre corecție și precizia cu care sunt determinați parametrii p este inferioară unui anumit număr (nu are rost să calculăm parametrii cu o precizie mult mai mică decât abaterea lor standard) .

Pentru a asigura convergența procesului de iterație , metoda Newton-Gauss a fost modificată în sensul introducerii unui parametru g în relația : $p^l = p^{l-1} + g\cdot\Delta p^l$, care face ca funcția χ^2 să depindă de acest parametru : $\chi^2 = \chi^2(g)$. Parametrul g se alege astfel încât parabola care trece prin punctele $(0,\chi^2(0))$, $(0.5,\chi^2(0.5))$, $(1,\chi^2(1))$ să admită o valoare minimă tocmai pentru această valoare g .

Observatie

In cazul în care funcția f(x,p) este chiar polinomul de gradul m-1 , relația (*) nu mai este doar aproximativă , ci este o relație exactă . In consecință , în urma rezolvării sistemului liniar , obținem nu o corecție aproximativă , ci chiar corecția adevărată . Mai mult , dacă vom lua ca soluție aproximativă inițială $p^0=(0,0,...,0)$, corecția va fi chiar soluția adevărată $\Delta p^1=p-p^0=p=(p_1,p_2,...,p_m)$. In acest caz obsevăm că sistemul de ecuații liniare devine identic cu cel de la fitarea cu un polinom , coeficienții matricilor A și B fiind aceiași .

6) Valorile medii ale lui χ^2

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i-f(x_i,p))^2}{\sigma_i^2} =$$

$$= \chi^2(x_1,..,x_n,y_1,..,y_n,p_1,..,p_m)$$

Deoarece χ^2 depinde de $y_1,..,y_n$, și aceste mărimi sunt supuse distribuției Gauss , putem defini media pe spațiul n-dimensional al acestor variabile :

$$\langle\chi^2(p)\rangle = \int_{(y_1)} .. \int_{(y_n)} \chi^2(x,y_1,..,y_n,p)\cdot$$

$$\cdot P(y_1)\cdot..\cdot P(y_n)\cdot dy_1\cdot..\cdot dy_n$$

Reamintim că :

$$\langle 1\rangle = \int_{(y_i)} P(y_i)\cdot dy_i = 1$$

$$<(y_i-\bar{y}_i)> = \int\limits_{(y_i)} (y_i-\bar{y}_i)\cdot P(y_i)\cdot dy_i = 0$$

$$<(y_i-\bar{y}_i)^2> = \int\limits_{(y_i)} (y_i-\bar{y}_i)^2\cdot P(y_i)\cdot dy_i = \sigma_i^2$$

Să calculăm media lui $\chi^2(\bar{p})$, pentru valorile exacte $\bar{p}$:

$$<\chi^2(\bar{p})> = \int\limits_{(y_1)} \cdots \int\limits_{(y_n)} \left(\sum_{i=1}^{n} \frac{(y_i-\bar{y}_i)^2}{\sigma_i^2}\right)\cdot$$

$$\cdot P(y_1)\cdot\ldots\cdot P(y_n)\cdot dy_1\cdot\ldots\cdot dy_n =$$

inversăm medierea cu sumarea :

$$= \sum_{i=1}^{n} \frac{1}{\sigma_i^2}\cdot \int\limits_{(y_1)} \cdots \int\limits_{(y_n)} (y_i-\bar{y}_i)^2\cdot$$

$$\cdot P(y_1)\cdot\ldots\cdot P(y_n)\cdot dy_1\cdot\ldots\cdot dy_n =$$

toate integralele dau 1 , în afară de cea pe (y_i) care dă σ_i^2 :

$$= \sum_{i=1}^{n} \frac{1}{\sigma_i^2}\cdot\sigma_i^2 = \sum_{i=1}^{n} 1 = n \qquad (7)$$

Să calculăm media lui $\chi^2(\tilde{p})$ pentru valorile probabile $\tilde{p}$, pentru cazul cel mai simplu al fitării cu o constantă :

$$\tilde{a} = \frac{\displaystyle\sum_{j=1}^{n} \frac{y_j}{\sigma_j^2}}{\displaystyle\sum_{j=1}^{n} \frac{1}{\sigma_j^2}}$$

$$<\chi^2(\tilde{a})> = <\sum_{i=1}^{n} \frac{(y_i-\tilde{a})^2}{\sigma_i^2}> =$$

$$= <\sum_{i=1}^{n} \frac{\left[y_i \displaystyle\sum_{j=1}^{n} \frac{1}{\sigma_j^2} - \sum_{j=1}^{n} \frac{y_j}{\sigma_j^2}\right]^2}{\left[\displaystyle\sum_{j=1}^{n} \frac{1}{\sigma_j^2}\right]^2 \cdot\sigma_i^2}> =$$

$$= \frac{1}{\left(\sum_{j=1}^{n} \frac{1}{\sigma_j^2}\right)^2} < \sum_{i=1}^{n} \frac{\left(\sum_{\substack{j=1\\j\neq i}}^{n} \frac{y_i-y_j}{\sigma_j^2}\right)^2}{\sigma_i^2} > =$$

$$= \frac{1}{\left(\sum_{j=1}^{n} \frac{1}{\sigma_j^2}\right)^2} < \sum_{i=1}^{n} \frac{\left(\sum_{\substack{j=1\\j\neq i}}^{n} \frac{y_i-\bar{a}}{\sigma_j^2} - \sum_{\substack{j=1\\j\neq i}}^{n} \frac{y_j-\bar{a}}{\sigma_j^2}\right)^2}{\sigma_i^2} > =$$

am realizat grupări (y_i-a) și (y_j-a) deoarece aplicând asupra acestora operația de mediere rezultatul este evident .

$$= \frac{1}{\left(\sum_{j=1}^{n} \frac{1}{\sigma_j^2}\right)^2} < \sum_{i=1}^{n} \frac{1}{\sigma_i^2}\cdot\left\{ \left(\sum_{\substack{j=1\\j\neq i}}^{n} \frac{y_i-\bar{a}}{\sigma_j^2}\right)^2 + \left(\sum_{\substack{j=1\\j\neq i}}^{n} \frac{y_j-\bar{a}}{\sigma_j^2}\right)^2 - \right.$$

$$\left. - 2\cdot\left(\sum_{\substack{j=1\\j\neq i}}^{n} \frac{y_i-\bar{a}}{\sigma_j^2}\right)\cdot\left(\sum_{\substack{j=1\\j\neq i}}^{n} \frac{y_j-\bar{a}}{\sigma_j^2}\right) \right\} > =$$

mediem cei trei termeni ai sumei după i :

în primul termen $(y_i-\bar{a})^2$ iese de sub suma după j , și dă σ_i^2

în al doilea termen produsele $(y_{j1}-\bar{a})(y_{j2}-\bar{a})$ prin mediere vor da 0 dacă $j1\neq j2$, sau vor da σ_j^2 dacă $j1=j2=j$

în al treilea termen $(y_i-\bar{a})$ iese de sub suma după j , și dă 0

$$= \frac{1}{\left(\sum_{j=1}^{n}\frac{1}{\sigma_j^2}\right)^2}\sum_{i=1}^{n}\frac{\sigma_i^2\cdot\left(\sum_{\substack{j=1\\j\neq i}}^{n}\frac{1}{\sigma_j^2}\right)^2+\sum_{\substack{j=1\\j\neq i}}^{n}\frac{\sigma_j^2}{\sigma_j^4}}{\sigma_i^2} =$$

$$= \frac{\sum_{i=1}^{n}\left[\left(\sum_{\substack{j=1\\j\neq i}}^{n}\frac{1}{\sigma_j^2}\right)\left(\left(\sum_{\substack{j=1\\j\neq i}}^{n}\frac{1}{\sigma_j^2}\right)+\frac{1}{\sigma_i^2}\right)\right]}{\left(\sum_{j=1}^{n}\frac{1}{\sigma_j^2}\right)^2} =$$

$$= \frac{\sum_{i=1}^{n}\left(\sum_{\substack{j=1\\j\neq i}}^{n}\frac{1}{\sigma_j^2}\right)}{\left(\sum_{j=1}^{n}\frac{1}{\sigma_j^2}\right)} = \frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\frac{1}{\sigma_j^2}-\frac{1}{\sigma_i^2}\right)}{\left(\sum_{j=1}^{n}\frac{1}{\sigma_j^2}\right)} =$$

$$= \left(\sum_{i=1}^{n}1\right) - 1 = n-1 \qquad (8)$$

In general , se presupune că funcția f(x) care descrie rezultatele experimentale depinde de $\bar{m}$ parametrii $\bar{p}$. Din condiția de minim a lui $\chi^2(p)$, exprimată pentru cele n puncte experimentale , vor rezulta parametrii cei mai probabili p , ce sunt diferiți de perametrii exacți $\bar{p}$. Am arătat că $<\chi^2(\bar{p})>=n$, și că în cazul în care m=1 , $<\chi^2(p)>=n-1$.
Se poate arăta pentru cazul general că $<\chi^2(\bar{p})>=n-m$. Valoarea $v=\bar{m}-n$ se numește gradul de libertate al funcției $\chi^2(\bar{p})$.

Pentru diverse seturi de n puncte experimentale , $\chi^2(\bar{p})$ va fi o variabilă aleatoare cu media v . In general se calculează $\chi^2(\bar{p})/v$ și se compară cu valoarea medie 1 . O valoare mai

mare de 1,5 indică următoarele posibilități :

a) punctele experimentale nu pot fi descrise de o funcție analitică de tipul celei utilizate în procesul de fitare

b) calculul fitării parametrilor nu a fost executat corect

c) abaterile standard ale măsurătorilor σ_i sunt mai mari în realitate decât cele utilizate în procesul de fitare

Deci valoarea lui $\chi^2(\bar{p})/v$ furnizează un crieriu de concordanță între rezultatele experimentale și ipotezele teoretice care încearcă să descrie fenomenul fizic studiat .

Observatie

In cazul în care toate erorile σ_i sunt egale cu σ , dar nu sunt cunoscute , din egalitatea aproximativă :

$$n-m = \langle\chi^2(\tilde{p})\rangle \approx \chi^2(\tilde{p}) = \frac{1}{\sigma^2}\sum_{i=1}^{n} (y_i-f(x_i,\tilde{p}))^2$$

se poate obține :

$$\sigma^2 = \frac{\sum_{i=1}^{n} (y_i-f(x_i,\tilde{p}))^2}{n-m} \qquad (9)$$

7) Determinarea abaterilor standard ale parametrilor fitati

Fie $\{y_i\}$, i=1..n , un spectru descris de funcția $f(x,\bar{p}_1,..,\bar{p}_m)$, cu parametrii $\bar{p}_j$ estimați prin metoda celor mai mici pătrate , y_i fiind mărimi aleatorii distribuite Gauss în jurul punctelor $f(x_i,p_1,..,p_m)$. In general $\bar{p}\neq p$. Repetând experiența în condiții identice se obțin alte valori y_i , deci alte valori $\bar{p}_j$. Se pune problema stabilirii funcției de distribuție a valorilor $\bar{p}$ obținute în urma repetării experienței .

Pentru ca distribuția Gauss a valorilor y_i în jurul valorilor $\bar{y}_i$ să determine o distribuție Gauss a valorilor $\bar{p}_j$ în jurul valorilor p_j , știm că este suficient ca parametrii $\bar{p}_j$ să depindă liniar de y_i . Pentru valorile $\bar{y}_i$, χ^2 se poate minimiza până la 0 , rezultând parametrii p_j . In acest caz vom putea folosi formula de propagare a erorilor pentru a afla abaterile standard ale parametrilor $\bar{p}$.

Să calculăm , pentru cazul cel mai simplu al fitării cu o constantă , dispersia valorii estimatei $\bar{a}$ în jurul valorii exacte a .

$$\tilde{a} = \frac{\sum_{j=1}^{n} \frac{y_j}{\sigma_j^2}}{\sum_{j=1}^{n} \frac{1}{\sigma_j^2}}$$

Avem dependență liniară de valorile y_i .

$$\sigma^2(\tilde{a}) = \sum_{i=1}^{n} \left(\frac{\partial \tilde{a}}{\partial y_i}\right)^2 \cdot \sigma_i^2 = \frac{\sum_{i=1}^{n} \frac{1}{\sigma_i^4} \cdot \sigma_i^2}{\left(\sum_{j=1}^{n} \frac{1}{\sigma_j^2}\right)^2} = \frac{1}{\sum_{j=1}^{n} \frac{1}{\sigma_j^2}}$$

In cazul în care $\sigma_j = \sigma$ pt orice j , avem :

$$\sigma^2(\tilde{a}) = \frac{\sigma^2}{n}$$

Pentru cazul general , parametrii rezultă din ultima rezolvare a sistemului corecțiilor (care în cazul fitării cu un polinom este și prima) :

$$\tilde{p}_k = p_k^l = p_k^{l-1} + \Delta p_k^l \qquad k=1..m$$

$$\Delta p_k^l = \frac{|D_k|}{|A|} \qquad |D_k| = \begin{vmatrix} a_{1,1} & \cdots & b_1 & \cdots & a_{1,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & \cdots & b_m & \cdots & a_{m,m} \end{vmatrix}$$

unde D_k provine din matricea A prin înlocuirea coloanei k cu coloana matricei B , iar matricile A și B sunt calculate pentru p^{l-1} (care în cazul fitării cu un polinom sunt nuli) .

Dependența lui $\tilde{p}_k$ de valorile y_i este în termenul Δp_k^l , și este liniară , deoarece valorile y_i apar numai în coloana k a determinantului $|D_k|$, la puterea întâi .

$$\sigma^2(\tilde{p}_k) = \sum_{i=1}^{n} \left(\frac{\partial \tilde{p}_k}{\partial y_i}\right)^2 \cdot \sigma_i^2 =$$

$$= \frac{1}{|A|^2} \cdot \sum_{i=1}^{n} \begin{vmatrix} a_{1,1} & \cdots & \partial b_1/\partial y_i & \cdots & a_{1,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m,1} & \cdots & \partial b_m/\partial y_i & \cdots & a_{m,m} \end{vmatrix}^2 \cdot \sigma_i^2 =$$

înlocuim pe $\frac{\partial b_j}{\partial y_i}$ cu $\frac{\partial f(x_i, p^{l-1})}{\partial p_j} \cdot \frac{1}{\sigma_i^2}$ și dezvoltăm determinantul după coloana k , notând cu $D_{j,k}$ complementul algebric al lui $A_{j,k}$:

$$= \frac{1}{|A|^2} \cdot \sum_{i=1}^{n} \left[\sum_{j=1}^{m} \frac{\partial f(x_i, p^{l-1})}{\partial p_j} \cdot \frac{1}{\sigma_i^2} \cdot D_{j,k} \right]^2 \cdot \sigma_i^2 =$$

$$= \frac{1}{|A|^2} \cdot \sum_{i=1}^{n} \left[\sum_{j=1}^{m} \frac{\partial f(x_i, p^{l-1})}{\partial p_j} \cdot D_{j,k} \right]^2 \cdot \frac{1}{\sigma_i^2} =$$

$$= \frac{1}{|A|^2} \cdot \sum_{i=1}^{n} \sum_{j,s=1}^{m} \frac{\partial f(x_i, p^{l-1})}{\partial p_j} \cdot \frac{\partial f(x_i, p^{l-1})}{\partial p_s} \cdot$$

$$\cdot D_{j,k} \cdot D_{s,k} \cdot \frac{1}{\sigma_i^2} =$$

$$= \frac{1}{|A|^2} \cdot \sum_{j,s=1}^{m} D_{j,k} \cdot D_{s,k} \cdot$$

$$\cdot \sum_{i=1}^{n} \frac{\partial f(x_i, p^{l-1})}{\partial p_j} \cdot \frac{\partial f(x_i, p^{l-1})}{\partial p_s} \cdot \frac{1}{\sigma_i^2} =$$

$$= \frac{1}{|A|^2} \cdot \sum_{j,s=1}^{m} D_{j,k} \cdot D_{s,k} \cdot A_{j,s} =$$

$$= \frac{1}{|A|^2} \cdot \sum_{s=1}^{m} D_{s,k} \cdot \sum_{j=1}^{m} D_{j,k} \cdot A_{j,s} =$$

suma după j reprezintă dezvoltarea după coloana k a unui determinant al unei matrici ce provine din matricea A prin înlocuirea coloanei k cu coloana s . In cazul în care k=s vom obține $|A|$, iar în cazul în care $k \neq s$ vom obține 0 (determinantul unei matrici cu două coloane egale, k și s , este nul) . Vom utiliza simbolul lui Kroneker δ_{ks} .

$$= \frac{1}{|A|^2} \cdot \sum_{s=1}^{m} D_{s,k} \cdot \delta_{ks} \cdot |A| = \frac{D_{k,k}}{|A|} = (A^{-1})_{k,k}$$

$$\sigma(\tilde{p}_k) = \sqrt{(A^{-1})_{k,k}} \qquad (10)$$

Astfel abaterea standard pentru parametrul p_k este radical de ordinul doi din al k-lea element diagonal al matricii A^{-1} . Matricea A^{-1} , inversa matricii sistemului corecțiilor , se mai numește și matricea erorilor .

8) Programul PASCAL

Programul prezentat în continuare folosește unit-ul sisec pentru rezolvarea sistemelor de ecuații liniare ale corecțiilor și aflarea matricii erorilor , și unit-ul graphics pentru reprezentarea grafică a punctelor și a funcției fitate . Unit-ul functie conține funcția cu care se fitează . Astfel , cu mici modificări în acest unit , se pot face fitări pentru diverse funcții. De asemenea sunt date și derivatele acestei funcții față de parametrii de care depinde .

Punctele se vor afla în fișiere de tipul :

```
XY      Y
x1      xi
y1      dx
x2      y1
y2      y2
...     ...
```

In al doilea caz , coordonatele pe axa x sunt la egală distanță dx , primul punct având coordonata xi . Pentru identificarea celor două cazuri , fișierele încep cu XY sau cu Y , urmând apoi valorile numerice .

Programul permite de asemenea ca datele măsurate să suporte o prelucrare primară , înainte de a deveni coordonate y . Acest lucru e util în calculul constantei de dezintegrare radioactivă , când pentru a liniariza funcția cu care se fitează (ca să ajungem la polinom de gradul I) , trebuie să logaritmăm numărul de dezintegrări (aflat pe fișierul de date) . In această prelucrare primară programul ține cont de legea de propagare a erorilor .

Erorile σ_d ale datelor măsurate d , înregistrate pe fișierul de date , sunt cerute sub forma $t \cdot d^q$, formă sub care putem introduce eroare constantă (q=0) , sau abaterea distribuției Poisson a dezintegrărilor radioactive (t=1 , q=0.5) . In cazul în care factorul t este necunoscut , putem introduce în locul lui caracterul '?' , programul urmând să-l determine , folosind relația :

$$n-m = \langle \chi^2(\tilde{p}) \rangle \approx \chi^2(\tilde{p})$$

Pentru identificarea cazului particular în care funcția de fitat e un polinom , caz în care rezolvarea sistemului corecțiilor se face doar o dată , corecțiile obținute fiind exacte , și nu aproximative , vom da constantei următoare valoarea : nume_functie='polinom'

Programul afișează rezultatele pe ecran , apoi face reprezentarea grafică , opțional acestea putînd fi scoase la imprimantă . Pentru a putea scoate graficul la imprimantă , prin apelarea în cod mașină a funcției PRINT SCREEN , va trebui , înainte de a lansa programul CMMP (din DOS sau

de sub TURBO PASCAL) , să executăm comanda GRAPHICS din DOS , care va permite scoaterea ecranului la imprimantă când lucrăm în modul grafic .

Iată în continuare programul , urmat de două unit-uri exemplificative pentru funcția fitată .

```
PROGRAM cmmp;
uses graph,crt,printer,sisec,graphics,functie;
type vector=array[1..100] of real;
var n,i:integer; {nr puncte experimentale,indice}
    x,y,s:vector;
    {matricile pt puncte si erori (sigma^2)}
    p,sp:mc;
    {matricile pt parametri si erori (sigma^2)}
    a,c:mp; b:mc;
    {matricile pt sist de ec al parametrilor}
    hi2n:real;
    {hi patrat normat la nr grd de lib}
    nume,sigmaY:string; t:real; ch:char;
    {variabile legate de citirea datelor}

FUNCTION PUTERE(baza,exponent:real):real;
begin
putere:=exp(exponent*ln(baza));
end;

PROCEDURE introd_date;    { INTRODUCEREA DATELOR }
var f:text; d,r,q,xi,dx:real; co,j:integer;
    legeY:char; ordine:string;
begin
clrscr;
writeln('CMMP  Estimare prin metoda ',
        'celor mai mici patrate');
writeln('functia cu care se fiteaza : ',
        nume_functie);
writeln('numarul de parametrii : ',m);
writeln;
write('Introd numele fisierului de date : ');
readln(nume);
writeln;
writeln('<1>   Y = d');
writeln('<2>   Y = d^r');
writeln('<3>   Y = ln(d)');
writeln('<4>   Y = exp(d)');
write('Introduceti coordonata Y ca functie ',
     'de data masurata d : ');
readln(legeY);
if legeY='2'
  then begin write('r = '); readln(r); end;
writeln;
writeln('Eroarea datei masurate este de forma ',
       'sigma(d) = t*d^q');
write('q = '); readln(q);
write('t = '); readln(sigmaY);
if sigmaY='?' then t:=1 else val(sigmaY,t,co);
writeln;
```

```
if nume_functie='polinom'
  then for j:=1 to m do p[j]:=0
  else begin
       writeln('Introd param aprox initiali');
       for j:=1 to m do begin
         write('p[',j,'] = '); readln(p[j]); end;
       writeln; end;
assign(f,nume); reset(f); n:=0;
readln(f,ordine);
if ordine='Y'
then begin readln(f,xi); readln(f,dx); end;
while not eof(f) do begin
   n:=n+1;
   if ordine='Y' then x[n]:=xi+(n-1)*dx;
   if ordine='XY' then readln(f,x[n]);
   readln(f,d);
   case legeY of
   '1': begin y[n]:=d; s[n]:=t*putere(d,q); end;
   '2': begin y[n]:=putere(d,r);
              s[n]:=r*t*putere(d,r+q-1); end;
   '3': begin y[n]:=ln(d);
              s[n]:=t*putere(d,q-1); end;
   '4': begin y[n]:=exp(d);
              s[n]:=t*putere(d,q)*exp(d); end;
   else halt; end;
   s[n]:=s[n]*s[n]; end;
close(f);
end;

FUNCTION hi2(p:mc):real;      { FUNCTIA HI PATRAT }
var h:real;
begin
h:=0;
for i:=1 to n do h:=h+sqr(y[i]-F(x[i],p))/s[i];
hi2:=h;
end;

PROCEDURE fit_F;              { FITAREA CU FUNCTIA }
var dp,p05,p1,er:mc; j,k,iter:integer;
    su,g,det,h0,h05,h1:real;
begin
iter:=0;
repeat
 for j:=1 to m do
 for k:=1 to m do
 begin
     su:=0;
     for i:=1 to n do
       su:=su+dF(x[i],p,j)*dF(x[i],p,k)/s[i];
     a[j,k]:=su;
 end;
 for j:=1 to m do
 begin
     su:=0;
     for i:=1 to n do
       su:=su+(y[i]-F(x[i],p))*dF(x[i],p,j)/s[i];
     b[j]:=su;
```

```
end;
sismat(m,a,b,det,dp,c);
if det=0 then begin writeln('det=0'); halt; end;
if nume_functie='polinom' then
  begin for j:=1 to m do
  begin p[j]:=p[j]+dp[j]; sp[j]:=c[j,j]; end;
  exit; end;
h0:=hi2(p);
for j:=1 to m do p05[j]:=p[j]+0.5*dp[j];
h05:=hi2(p05);
for j:=1 to m do p1[j]:=p[j]+dp[j];
h1:=hi2(p1);
g:=(h1+3*h0-4*h05)/(4*(h1+h0-2*h05));
if (g<0) or (g>1.5) then g:=1;
for j:=1 to m do begin
  p[j]:=p[j]+g*dp[j]; sp[j]:=g*g*c[j,j]; end;
for j:=1 to m do er[j]:=abs(dp[j])/sqrt(sp[j]);
iter:=iter+1;
until (maxim(m,er)<0.1) or (iter=100);
writeln('nr iteratii este : ',iter); writeln;
end;

PROCEDURE test_hi2;        { VERIFICAREA HI_PATRAT }
var h:real;
begin
h:=hi2(p)/(n-m);
if sigmaY='?'
     then begin
          for i:=1 to n do s[i]:=h*s[i];
          t:=sqrt(h); end
     else hi2n:=h;
end;

PROCEDURE rezultate(var device:text);
                          { AFISAREA SAU IMPRIMAREA }
var j:integer;
begin
writeln(device,'CMMP  Estimare prin metoda ',
            'celor mai mici patrate');
writeln(device,'functia cu care se fiteaza : ',
            nume_functie);
writeln(device,'numarul de parametrii : ',m);
writeln(device,'fisier de date : ',nume);
writeln(device);
if sigmaY='?'
then writeln(device,'t = ',t)
else writeln(device,'(hi^2)normat = ',hi2n);
writeln(device);
writeln(device,'parametrii rezultati in urma ',
            'fitului : ');
for j:=1 to m do
writeln(device,'p[',j,'] = ',p[j],
            ' +/- ',sqrt(sp[j]));
writeln(device);
end;

PROCEDURE desen;           { REPREZENTAREA GRAFICA }
```

```
var x1,x2,y1,y2,d,dx,xr:real;
begin
x1:=x[1]; x2:=x[1]; y1:=y[1]; y2:=y[1];
for i:=2 to n do begin
   if x[i]<x1 then x1:=x[i];
   if x[i]>x2 then x2:=x[i];
   if y[i]<y1 then y1:=y[i];
   if y[i]>y2 then y2:=y[i]; end;
d:=x2-x1; x1:=x1-d/10; x2:=x2+d/10;
d:=y2-y1; y1:=y1-d/10; y2:=y2+d/10;
init_graph_mode;
init_window(0,maxx,0,maxy,x1,x2,y1,y2);
for i:=1 to n do star_err(x[i],y[i],sqrt(s[i]));
                                   {PUNCTELE SI ERORILE}
dx:=1/scaraX; xr:=XminR;          {FUNCTIA FITATA}
repeat
      putpixel(xe(xr),ye(F(xr,p)),white);
      xr:=xr+dx;
until xr>XmaxR;
end;

{ ########## INCEPUTUL PROGRAMULUI ############ }
begin
clrscr;
introd_date;
clrscr;
fit_F;
test_hi2;
rezultate(output);
writeln('Scoaterea parametrilor la imprimanta ',
      '          y/n');
ch:=readkey; if ch='y' then rezultate(lst);
writeln('Scoaterea graficului la imprimanta ',
      '            y/n');
ch:=readkey;
desen;
if ch='y' then inline($CD/$05); { Print Screen }
repeat until readkey='q'; closegraph;
end.


UNIT FUNCTIE;

interface

uses sisec;
const m=2; {= numarul de parametrii = gradul + 1}
      nume_functie='polinom'; {de grad m - 1 = 1}
function F(x:real;p:mc):real;
function dF(x:real;p:mc;j:integer):real;

implementation

FUNCTION F;
var v:real; i:integer;
begin
     v:=p[m];
```

```
    for i:=m-1 downto 1 do v:=v*x+p[i];
    F:=v;
end;

FUNCTION dF;
var v:real; i:integer;
begin
    v:=1;
    for i:=1 to j-1 do v:=v*x;
    dF:=v;                       { x la puterea j-1 }
end;

end.
```

```
UNIT FUNCTIE;

interface

uses sisec;
const m=2; { = numarul de parametrii }
    nume_functie='p[1]+p[2]/x'; { hiperbola }
function F(x:real;p:mc):real;
function dF(x:real;p:mc;j:integer):real;

implementation

FUNCTION F;
begin
    F:=p[1]+p[2]/x;
end;

FUNCTION dF;
begin
    case j of
    1: dF:=1;
    2: dF:=1/x; end;
end;

end.
```

9) Problema

In laboratorul de Fizică Nucleară , studiind legea de activare , s-a măsurat activitatea vanadiului , numărând dezintegrările radioactive din intervale succesive de 10 s . După ce s-a scăzut fondul , iată numărul de dezintegrări :

165,172,150,129,156,141,139,134,123,139,118, 121,105,108,111,105,96,88,116,101,96,93,70,71, 64,75,60,70,73,63,62,63,60,69,57,35,53,46,55,61, 57,48,39,32,26,37,40,44,38,45,34,37,27,38,35,22, 27,17,26,27,24,18,16,25,15,26,15,17,16,18

Legea dezintegrării radioactive fiind $n(t)=n(0)e^{-\lambda t}$, prin logaritmare obținem

$\ln(n(t))=\ln(n(0))-\lambda t$.

1) Alcătuiți un fișier cu aceste date . El va începe cu liniile :

Y / 10 / 10 / 165 / 172 / ...

2) Fitați datele cu un polinom de rang I , logaritmând datele măsurate . Procedați în două moduri :

a) Selectați doar datele până la valoarea 48 inclusiv

b) Prelucrați toate datele

Comparați pentru cele două cazuri valorile lui χ^2normat . Care este mai aproape de 1 ? Comparați constantele de dezintegrare obținute cu cea luată din tabel :

$\lambda_{vanadiu} = 3.064\cdot10^{-3}s^{-1}$

Dați o explicație .

Raspuns

Pentru a putea aproxima distribuția Poisson a dezintegrărilor radioactive cu distribuția Gauss folosită de către programul de fit trebuie să ne oprim la valorile mai mari de 50 .

CAP XII. GASIREA RADACINILOR REALE SI COMPLEXE ALE UNUI POLINOM

1) Algoritmul

Fie un polinom , cu coeficienți reali , de variabilă complexă z :

$$P(z) = c_n \cdot z^n + .. + c_1 \cdot z + c_0 = $$
$$= Re(P(z)) + i \cdot Im(P(z)) \qquad (1)$$

Modulul acestui polinom este un număr pozitiv, care ia însă valoarea zero atunci când z este o rădăcină a polinomului .

$$|P(z)| = \sqrt{Re(P(z))^2 + Im(P(z))^2}$$
$$|P(z)| = 0 \Leftrightarrow Re(P(z)) = Im(P(z)) = 0 \Leftrightarrow P(z) = 0$$

Vom căuta minimele lui |P(z)| , care sunt totodată și rădăcinile polinomului . Metoda de căutare este următoarea : pornim cu o valoare inițială pentru z , z = a + ib , și variem pe rând pe a și pe b , cu pașii da și db , astfel ca întotdeauna modulul polinomului să scadă .

Vom arăta cum facem alegerea pentru variația lui a , pentru b fiind același lucru .

Mai întâi calculăm |P(a,b)|,|P(a+da,b)|, |P(a-da,b)| . In cazul în care |P(a,b)| este mai mic decât celelalte două module , înseamnă că între a-da și a+da se află minimul căutat . In acest caz înjumătățim pasul da . Dacă astfel pasul da a ajuns mai mic decât precizia dorită , minimul față de a a fost localizat . Dacă pasul da este încă suficient de mare , atunci la următoarea iterație se vor analiza punctele a-da/2 , a , a+da/2 , având pasul da/2 . Deoarece pornind de la aceste puncte , cu pasul da/2 , acoperim întregul interval (a-da,a+da) în care știm că se află minimul , avem certitudinea că micșorarea pasului nu ne poate duce la ocolirea minimului căutat .

In cazul în care |P(a+da)| este cel mai mic , îl vom crește pe a cu da . In cazul în care |P(a-da)| este cel mai mic , îl vom micșora pe a cu da .

Repetăm procedeul , variind pe a și pe b , până când atât da cât și db au ajuns suficient de mici , astfel încât să avem precizia dorită .

O dată ce știm o rădăcină , verificăm dacă aceasta este reală sau complexă . In caz că este reală , împărțim polinomul la (z-a-ib) . In caz că este complexă , știm că și conjugata sa a-ib este rădăcină , și împărțim polinomul la (z-a-ib)(z-a+ib) . Prin repetarea procedeului gradul polinomului scade până la zero , moment în care toate rădăcinile lui , reale sau complexe , au fost determinate .

2) Programul PASCAL

Programul a fost scris sub forma unei proceduri dintr-un unit , astfel încât să fie ușor de folosit de către alte programe . Alte proceduri ale acestui unit , folosite pentru determinarea rădăcinilor polinomului , sunt : val_polinom ce calculează valoarea unui polinom într-un punt dat , modul_polinom ce calculează modulul unui polinom într-un punct dat , imp_monom ce efectuează reducerea gradului polinomului în cazul în care rădăcina găsită este reală , imp_binom ce efectuează reducerea gradului polinomului în cazul în care rădăcina găsită este complexă .

Iată în contiuare unit-ul , cât și un program exemplificativ care îl folosește :

```
UNIT POLINOM;

interface

const eps=1e-10;
type grad=0..10;
     vector=array[grad] of real;
     { matrice pt coeficienti polinom }
     sol=record
         re,im:real;
         end;
     vsol=array[1..10] of sol;
     { matrice pt solutiile polinomului }
procedure val_polinom(c:vector; n:grad;
                      a,b:real; var s:sol);
function modul_polinom(c:vector; n:grad;
                      a,b:real):real; { z=a+ib }
procedure sol_polinom(c:vector; n:grad;
                      var vs:vsol);

implementation

PROCEDURE VAL_POLINOM;
var i:grad; pr,pi,zr,zi,zrn,zin:real;
begin
pr:=0; pi:=0; zr:=1; zi:=0;
for i:=0 to n do begin
{se aduna contributiile termenilor z^n}
   pr:=pr+zr*c[i]; pi:=pi+zi*c[i];
   {realul si imaginarul polinomului}
   zrn:=zr*a-zi*b; zin:=zr*b+zi*a;
   {realul si imaginarul lui z^(n+1)}
   zr:=zrn; zi:=zin;
   {se trece la puterea urmatoare a lui z}
   end;
s.re:=pr; s.im:=pi;
end;
```

```
FUNCTION MODUL_POLINOM;
var s:sol;
begin
val_polinom(c,n,a,b,s);
modul_polinom:=sqrt(s.re*s.re+s.im*s.im);
end;

procedure imp_monom(var c:vector; var n:grad;
                    a:real); { Horner }
var i:grad;
begin { impartire la monomul (x-a) }
for i:=n-1 downto 1 do c[i]:=c[i]+a*c[i+1];
{ in final c[0]=0=restul }
for i:=0 to n-1 do c[i]:=c[i+1];
n:=n-1;
end;

procedure imp_binom(var c:vector; var n:grad;
                    a,b:real);
var i:grad;
begin { impartire la binomul (x-a-ib)(x-a+ib) }
for i:=n downto 2 do begin
{ in final c[0]=c[1]=0=restul }
   c[i-1]:=c[i-1]+2*a*c[i];
   c[i-2]:=c[i-2]-(a*a+b*b)*c[i]; end;
for i:=0 to n-2 do c[i]:=c[i+2];
n:=n-2;
end;

PROCEDURE SOL_POLINOM;
var a,b,da,db:real; p,m,z:real;
begin
repeat { pentru cele n solutii }
   a:=0; b:=0; { aproximatia initiala }
   da:=1; db:=1;{pasii initiali pe cele doua axe}
   repeat { pina cind precizia sol e suficienta }
      { testarea variatiei lui a }
      z:=modul_polinom(c,n,a,b);
      p:=modul_polinom(c,n,a+da,b);
      m:=modul_polinom(c,n,a-da,b);
      if p<z then a:=a+da
             else if m<z then a:=a-da
                         else if da>eps
                              then da:=da/2;
      { testarea variatiei lui b }
      z:=modul_polinom(c,n,a,b);
      p:=modul_polinom(c,n,a,b+db);
      m:=modul_polinom(c,n,a,b-db);
      if p<z then b:=b+db
             else if m<z then b:=b-db
                         else if db>eps
                              then db:=db/2;
   until (da<=eps) and (db<=eps);
   if abs(b)<=eps
   then begin vs[n].re:=a; vs[n].im:=0;
        imp_monom(c,n,a); end
   else begin vs[n].re:=a; vs[n].im:=b;
```

```
        vs[n-1].re:=a; vs[n-1].im:=-b;
        imp_binom(c,n,a,b); end;
until n=0;
end;

end.


PROGRAM usepol;
uses polinom;
var c:vector; n,i:grad; vs:vsol;
begin
write('Gradul polinomului este :  n = ');
readln(n);
writeln('Introduceti coeficientii polinomului');
for i:=n downto 0 do begin
   write('c[',i,'] = '); readln(c[i]); end;
sol_polinom(c,n,vs);
writeln('Solutiile polinomului sunt');
for i:=1 to n do
  writeln(i,'   re=',vs[i].re,'   im=',vs[i].im);
readln;
end.
```

CAP XIII. VALORI SI VECTORI PROPRII

1) Polinomul caracteristic

In multe probleme de fizică se obține o ecuație de forma :

$$AX=\lambda X \qquad (1)$$

unde A este o matrice pătratică , λ este un număr , numit și valoare proprie a matricii A , iar X este o matrice coloană , numit și vector propriu al matricii A , corespunzător valorii proprii λ .

Ecuația (1) poate fi pusă sub forma :

$$(A-\lambda I)X=0 \qquad (2)$$

unde I este matricea unitate .

Pentru a avea $X\neq0$ trebuie ca determinantul să se anuleze :

$$\det(A-\lambda I)=0 \qquad (3)$$

Termenul din stânga al ecuației (3) este un polinom în λ , numit polinom caracteristic .

Odată găsite rădăcinile polinomului caracteristic , prin rezolvarea sistemului nedeterminat de ecuații liniare se obțin cu ușurință și vectorii proprii .

Găsirea rădăcinilor unui polinom a fost tratată mai devreme . Tot ce trebuie să facem este să obținem coeficienții polinomului caracteristic . Pentru aceasta există mai multe metode .

2) Metoda coeficientilor nedeterminati

Să notăm :

$$\det(\lambda I-A) = D(\lambda) =$$
$$= \lambda^n + p_1\cdot\lambda^{n-1} + \ldots + p_{n-1}\cdot\lambda + p_n \qquad (4)$$

Coeficienții nedeterminați ai polinomului caracteristic sunt p_1 , .. , p_n .

Noi putem să calculăm numeric valoarea determinantului $D(\lambda)$ pentru o valoare dată a lui λ , cu ajutorul algoritmului de diagonalizare al lui Gauss . Vom da lui λ un număr de n valori particulare , și astfel vom obține un sistem de n ecuații liniare , cu necunoscutele chiar coeficienții căutați . Rezolvarea acestui sistem de ecuații liniare se face tot cu ajutorul metodei lui Gauss .

Vom avea sistemul :

$$p_1\cdot\lambda_1^{n-1} + .. + p_n = \det(\lambda_1 I-A) - \lambda_1^n \qquad (5)$$
$$\ldots$$
$$p_1\cdot\lambda_n^{n-1} + .. + p_n = \det(\lambda_n I-A) - \lambda_n^n$$

Observând că pentru valoarea particulară $\lambda=0$ ecuația rezultantă se reduce la $p_n=\det(A)$, dacă alegem setul de valori particulare :

$\lambda = 0,1,..,n-1$

putem să reducem numărul de ecuații al

sistemului coeficienților nedeterminați de la n la n-1 .

Vom avea sistemul :

$$p_1 + p_2 + .. + p_{n-1} = D(1) - D(0) - 1$$
$$2^{n-1} \cdot p_1 + 2^{n-2} \cdot p_2 + .. + 2 \cdot p_{n-1} = D(2) - D(0) - 2^n$$
....
$$(n-1)^{n-1} \cdot p_1 + (n-1)^{n-2} \cdot p_2 + .. + (n-1) \cdot p_{n-1} = D(n-1) - D(0) - (n-1)^n$$

In cazul în care trebuie să calculăm valorile proprii pentru mai multe matrici , este convenabil să utilizăm o matrice comună :

$$C = \begin{pmatrix} 1 & 1 & .. & 1 \\ 2^{n-1} & 2^{n-2} & .. & 2 \\ .... & & & \\ (n-1)^{n-1} & (n-1)^{n-2} & .. & (n-1) \end{pmatrix} \quad (6)$$

noi făcând calculele doar pentru termenii liberi ai fiecăruia din sistemele de ecuații ale coeficienților :

$$D = \begin{pmatrix} D(1)-D(0)-1 \\ D(2)-D(0)-2^n \\ ..... \\ D(n-1)-D(0)-(n-1)^n \end{pmatrix} \quad (7)$$

și rezolvând apoi sistemul de ecuații liniare CP=D , cu :

$$P = \begin{pmatrix} p_1 \\ p_2 \\ .. \\ p_{n-1} \end{pmatrix} \quad (8)$$

3) Metoda Leverrier

Fie :

$$\det(\lambda I-A) = \lambda^n + p_1 \cdot \lambda^{n-1} + ... + p_{n-1} \cdot \lambda + p_n \quad (9)$$

Să notăm rădăcinile polinomului caracteristic cu λ_1 , .. , λ_n , și

$$s_k = \lambda_1^k + \lambda_2^k + .. + \lambda_n^k \quad (10)$$

Ne vom folosi de faptul că dacă λ_1 , .. , λ_n sunt valorile proprii ale lui A , atunci :

$$\lambda_1 + \lambda_2 + .. + \lambda_n = \mathrm{trace}(A) \quad (11)$$

unde trace(A) este suma elementelor diagonale ale matricii A .

Intr-adevăr :

$$\begin{vmatrix} \lambda-a_{11} & -a_{12} & .. & -a_{1n} \\ -a_{21} & \lambda-a_{22} & .. & -a_{2n} \\ ..... & & & \\ -a_{n1} & -a_{n2} & .. & \lambda-a_{nn} \end{vmatrix} =$$

$$= \lambda^n - \lambda^{n-1} \cdot (a_{11} + a_{22} + .. + a_{nn}) + .. =$$
$$= (\lambda-\lambda_1) \cdot (\lambda-\lambda_2) \cdot .. \cdot (\lambda-\lambda_n) =$$

$= \lambda^n - \lambda^{n-1} \cdot (\lambda_1 + \lambda_2 + .. + \lambda_n) + ..$

și prin identificarea coeficientului lui λ^{n-1} rezultă (11) .

Astfel l-am obținut pe s_1 . Pentru a-i obține și pe restul s_k , cu k=2..n , trebuie să calculăm urmele unor matrici care au ca valori proprii pe λ_1^k , λ_2^k , .. , λ_n^k . Acestea vor fi tocmai puterile k ale matricii A .

Intr-adevăr :

$$AX=\lambda X \Rightarrow A^2X=\lambda AX=\lambda^2X \Rightarrow .. \Rightarrow A^nX=\lambda^nX \quad (12)$$

Odată cunoscute sumele s_k , cu k=1..n , putem obține coeficienții p_1 , .. , p_n cu ajutorul formulei lui Newton , care spune că având un polinom de forma (9) , ce are rădăcinile λ_1 , .. , λ_n , cu notația din (10) vom avea :

$$s_k + p_1 \cdot s_{k-1} + .. + p_{k-1} \cdot s_1 = -k \cdot p_k \quad (13)$$

Această formulă se demonstrează ușor dacă înlocuim toți termenii cu expresiile lor funcție de λ_1 , .. , λ_n , adică pe s_k din formula (10) și pe p_k din formulele lui Viète .

Reamintin că formulele lui Viète se deduc , prin identificarea coeficienților puterilor lui λ , din :

$$\lambda^n + p_1 \cdot \lambda^{n-1} + ... + p_{n-1} \cdot \lambda + p_n =$$
$$= (\lambda-\lambda_1) \cdot (\lambda-\lambda_2) \cdot .. \cdot (\lambda-\lambda_n)$$

rezultând :

$$p_k = (-1)^k \cdot \sum_{\substack{i_1,..,i_k \in 1..n \\ \text{diferiți}}} \lambda_{i_1} \cdot .. \cdot \lambda_{i_k} \quad (14)$$

Coeficienții polinomului caracteristic sunt obținuți sub forma :

$$p_1 = -s_1$$

$$p_2 = -\frac{1}{2} \cdot (s_2 + p_1 \cdot s_1)$$

...

$$p_n = -\frac{1}{n} \cdot (s_n + p_1 \cdot s_{n-1} + .. + p_{n-1} \cdot s_1)$$

4) Programele PASCAL

In continuare sunt date două programe ce determină valorile proprii ale unei matrici .

Primul folosește metoda coeficienților nedeterminați , dar pentru a fi ușor de înțeles s-a renunțat la reducerea numărului de ecuații cu una prin folosirea valorii particulare $\lambda=0$, programul folosind sistemul (5) , cu valorile particulare $\lambda=1..n$.

Al doilea folosește metoda Leverrier .

Ambele programe folosesc unit-ul polinom.pas , pentru a afla rădăcinile polinomului caracteristic . Deoarece acest unit folosește o altă convenție de notare a coeficienților unui

polinom :

$$P(x) = p_n \cdot x^n + p_{n-1} \cdot x^{n-1} + .. + p_1 \cdot x + p_0$$

este necesară o prealabilă trecere de la o notație la cealaltă .

```
PROGRAM coef_ned;       { val proprii prin metoda }
uses polinom,sisec;     { coef nedeterminati }
var m,p2,a,r:mp; p1,b,s:mc; n,i,j:dec;
    p:vector; vs:vsol; det:real;

FUNCTION PUTERE(baza:real;exponent:integer):real;
var i:integer; s:real;
begin
s:=1;
if exponent<0
then for i:=-1 downto exponent do s:=s/baza;
if exponent>0
then for i:=1 to exponent do s:=s*baza;
putere:=s;
end;

begin
{ CITIREA MATRICII CAREIA II CAUTAM VAL PROPRII }
write('Ordinul matricii (1..10) :  n = ');
readln(n);
writeln('Introduceti elementele matricii');
for i:=1 to n do
for j:=1 to n do begin
   write('m(',i,',',j,') = ');
   readln(m[i,j]); end;
{ FORMAREA SIST DE EC PT COEF POLIN CARACT }
for i:=1 to n do begin
{ pentru n valori particulare ale lui lambda }
   a:=m;
   for j:=1 to n do a[j,j]:=a[j,j]-i;  {lambda=i}
   sismat(n,a,b,det,s,r);
   { det=|A-iI| , noi folosim |iI-A| }
   p1[i]:=putere(-1,n)*det-putere(i,n); end;
for i:=1 to n do
for j:=1 to n do p2[i,j]:=putere(i,n-j);
{ AFLAREA COEFICIENTILOR POLIN CARACTERISTIC }
sismat(n,p2,p1,det,s,r);
for i:=1 to n do p[n-i]:=s[i]; p[n]:=1;
{ AFLAREA RADACINILOR POLINOMULUI CARACTERISTIC }
sol_polinom(p,n,vs);
writeln('Valorile proprii sunt');
for i:=1 to n do
  writeln(i,'   re=',vs[i].re,'   im=',vs[i].im);
readln;
end.
```

```
PROGRAM Leverrier; { determinarea val pr }
uses polinom;          { prin metoda Leverrier }
type mat=array[1..10,1..10] of real;
var a,b:mat; i,j,n:integer; sp:real;
```

```
    p:array[1..10] of real; { coef polin caract }
    s:array[1..10] of real; { sumele radacinilor}
    pv:vector; vs:vsol;

PROCEDURE prod_mat(a,b:mat;n:integer;var c:mat);
var i,j,k:integer; s:real;
begin
     for i:=1 to n do
     for j:=1 to n do begin
        s:=0;
        for k:=1 to n do s:=s+a[i,k]*b[k,j];
        c[i,j]:=s; end;
end;

FUNCTION trace(a:mat;n:integer):real;
var i:integer; s:real;
begin
     s:=0;
     for i:=1 to n do s:=s+a[i,i];
     trace:=s;
end;

begin
{ CITIREA MATRICII CAREIA II CAUTAM VAL PROPRII }
write('Ordinul matricii (1..10) :  n = ');
readln(n);
writeln('Introduceti elementele matricii');
for i:=1 to n do
for j:=1 to n do begin
   write('a(',i,',',j,') = ');
   readln(a[i,j]); end;
{ CALCULAREA SUMELOR RADACINILOR LA PUTERE }
b:=a;
for i:=1 to n do begin
   s[i]:=trace(b,n);
   prod_mat(a,b,n,b); end;
{ CALCULAREA COEFICIENTILOR POLINOMULUI CARACT }
for i:=1 to n do begin
   sp:=0;
   for j:=1 to i-1 do sp:=sp+s[j]*p[i-j];
   sp:=sp+s[i];
   p[i]:=sp/(-i); end;
{ AFLAREA RADACINILOR POLINOMULUI CARACTERISTIC }
pv[n]:=1;
for i:=0 to n-1 do pv[i]:=p[n-i];
sol_polinom(pv,n,vs);
writeln('Valorile proprii sunt');
for i:=1 to n do
  writeln(i,'   re=',vs[i].re,'   im=',vs[i].im);
readln;
end.
```

5) Probleme

1) Alcătuiți un program care să determine valorile proprii ale unei matrici prin metoda

coeficienţilor nedeterminaţi , dar care să reducă numărul de ecuaţii cu una prin folosirea valorii particulare $\lambda=0$.

2) Demonstraţi formula lui Newton (13) .

CAP XIV. DETERMINAREA VALORILOR SI VECTORILOR PROPRII AI UNEI MATRICI SIMETRICE REALE PRIN METODA JACOBI

1) Notiuni despre spatii vectoriale

Fie un spațiu vectorial complex de dimensiune n , și o bază :

$$e_1 , e_2 , .. , e_n \quad (1)$$

Orice vector al spațiului vectorial poate fi descompus în mod unic după vectorii bazei :

$$\underline{x} = x_1 \cdot e_1 + x_2 \cdot e_2 + .. + x_n \cdot e_n = (x_1 , x_2 , .. , x_n) \quad (2)$$

Valorile complexe $x_1 , x_2 , .. , x_n$ se numesc coordonatele vectorului $\underline{x}$ în baza (1) .

Fie un alt vector :

$$\underline{y} = y_1 \cdot e_1 + y_2 \cdot e_2 + .. + y_n \cdot e_n = (y_1 , y_2 , .. , y_n) \quad (3)$$

Produsul scalar dintre cei doi vectori este definit astfel :

$$(\underline{x} , \underline{y}) = \sum_{i=1}^{n} x_i \cdot y_i^* \quad (4)$$

unde * semnifică operația de conjugare complexă .

Produsul scalar are următoarele proprietăți :

1) $(\underline{x} , \underline{x}) \geq 0$, fiind egal cu zero doar pentru $\underline{x} = 0$

2) $(\underline{x} , \underline{y}) = (\underline{y} , \underline{x})^*$

3) $(\alpha \cdot \underline{x} , \underline{y}) = \alpha \cdot (\underline{x} , \underline{y})$

$(\underline{x} , \alpha \cdot \underline{y}) = \alpha^* \cdot (\underline{x} , \underline{y})$

Lungimea vectorului $\underline{x}$ este definită astfel :

$$|\underline{x}| = \sqrt{(\underline{x} , \underline{x})} \quad (5)$$

fiind un număr nenegativ .

Se spune că vectorii $\underline{x}$ și $\underline{y}$ sunt ortogonali dacă produsul lor scalar este nul :

$$(\underline{x} , \underline{y}) = 0 \quad (6)$$

Baza (1) se spune că este ortonormată dacă :

$$(e_i , e_j) = \delta_{ij}$$

unde am introdus simbolul lui Kroneker :

$$\delta_{ij} = \begin{cases} 1 , & i=j \\ 0 , & i \neq j \end{cases}$$

Fie un vector $\underline{x}$, și un alt vector $\underline{y}$ ale cărui coordonate rezultă din combinații liniare ale coordonatelor primului vector :

$$y_i = \sum_{j=1}^{n} a_{ij} \cdot x_j \quad (7)$$

Notând :

$$x = \begin{pmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{pmatrix} \qquad y = \begin{pmatrix} y_1 \\ y_2 \\ .. \\ y_n \end{pmatrix} \qquad (8)$$

vom avea :

$$y = A\,x \qquad (9)$$

unde A este matricea (a_{ij}) .

Spunem că A este matricea unui operator liniar , deoarece pentru orice x , z matrici coloană ale coeficienților vectorilor , și pentru orice α , β numere complexe vom avea :

$$A\,(\,\alpha\cdot x + \beta\cdot z\,) = \alpha\cdot A\,x + \beta\cdot A\,z \qquad (10)$$

Operația de compunere a doi operatori liniari se reduce la operația de înmulțire a matricilor celor doi operatori .

2) Schimbarea de baza

Fie o altă bază a spațiului vectorial :

$$\varepsilon_1\,,\ \varepsilon_2\,,\ ..\,,\ \varepsilon_n \qquad (11)$$

Vectorii bazei (11) se pot descompune după vectorii bazei (1) :

$$\varepsilon_j = \sum_{i=1}^{n} s_{ij}\cdot e_i \qquad (12)$$

Matricea $S = (s_{ij})$ se numește matricea de trecere de la o bază la alta .

Notând :

$$e = \begin{pmatrix} e_1 \\ e_2 \\ .. \\ e_n \end{pmatrix} \qquad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ .. \\ \varepsilon_n \end{pmatrix} \qquad (13)$$

vom avea :

$$\varepsilon = S^T e \qquad (14)$$

unde S^T este transpusa matricii S .

In cazul în care cele două baze sunt ortonormate :

$$(\,e_i\,,\,e_j\,) = \delta_{ij} \qquad (\,\varepsilon_i\,,\,\varepsilon_j\,) = \delta_{ij} \qquad (15)$$

folosind (12) vom avea :

$$\delta_{ij} = (\,\varepsilon_i\,,\,\varepsilon_j\,) = (\,\sum_{k=1}^{n} s_{ki}\cdot e_k\,,\ \sum_{r=1}^{n} s_{rj}\cdot e_r\,) =$$

$$= \sum_{k=1}^{n} s_{ki} \sum_{r=1}^{n} s_{rj}\,(\,e_k\,,\,e_r\,) = \sum_{k=1}^{n} s_{ki} \sum_{r=1}^{n} s_{rj}\,\delta_{kr} =$$

$$= \sum_{k=1}^{n} s_{ki}\,s_{kj}$$

Relația de mai sus se poate scrie matricial sub forma :

$$S^T S = I \qquad (16)$$

unde am notat prin I matricea unitate .

O matrice a cărei transpusă este chiar inversa ei se numește matrice ortogonală . Deci , comform relației (16) , matricea de trecere de la o bază ortonormată la alta este ortogonală .

3) Transformarea coordonatelor vectorilor

Fie vectorul $\underline{x}$ descompus după două baze :

$$\underline{x} = \sum_{i=1}^{n} x_i \cdot e_i = \sum_{j=1}^{n} \xi_j \cdot \varepsilon_j \qquad (17)$$

Folosim formula (12) pentru eliminarea lui ε_j :

$$\sum_{i=1}^{n} x_i \cdot e_i = \sum_{j=1}^{n} \xi_j \sum_{i=1}^{n} s_{ij} \cdot e_i = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \xi_j \cdot s_{ij} \right) \cdot e_i$$

Ținând cont de liniar independența vectorilor bazei , obținem :

$$x_i = \sum_{j=1}^{n} \xi_j \cdot s_{ij} \qquad (18)$$

Notând :

$$x = \begin{pmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{pmatrix} \qquad \xi = \begin{pmatrix} \xi_1 \\ \xi_2 \\ .. \\ \xi_n \end{pmatrix} \qquad (19)$$

vom avea :

$$x = S\ \xi \qquad (20)$$

Determinantul matricii S este nenul , deoarece dacă ar fi nul ar rezulta că o coloană ar putea să fie scrisă ca o combinație liniară a celorlalte , și atunci , comform relațiilor (12) , ar rezulta că un vector al bazei ε ar putea să fie scris ca o combinație liniară a celorlalți , ceea ce este absurd . In consecință matricea S poate fi inversată , și din (20) obținem :

$$\xi = S^{-1} x \qquad (21)$$

4) Transformarea matricilor operatorilor liniari

Fie matricile coloană ale componentelor în baza e ale vectorilor $\underline{x}$ și $\underline{y}$, legate prin matricea operatorului liniar A :

$$y = A\ x \qquad (22)$$

La o transformare a bazei din e în ε , componentele vectorilor se vor schimba :

$$x = S\ \xi \qquad y = S\ \zeta \qquad (23)$$

Relația (22) se va scrie :

$$S\ \zeta = A\ S\ \xi$$

Prin înmulțirea la stânga cu S^{-1} obținem :

$\zeta = S^{-1} A S \xi$

$\zeta = B \xi$ (24)

Deci matricea operatorului liniar A , la o schimbare de bază , trece în :

$B = S^{-1} A S$ (25)

5) Vectorii si valorile proprii ai unei matrici

Fie A matricea unui operator liniar , și x și A x matricile coloană ale coordonatelor unor vectori .

Spunem că numărul complex λ este o valoare proprie a lui A , și că x este un vector propriu al lui A , corespunzător valorii proprii λ , dacă , pentru $\lambda \neq 0$ și $x \neq 0$, avem :

$A x = \lambda \cdot x$ (26)

Rezultă :

$(A - \lambda \cdot I) x = 0$ (27)

Pentru ca sistemul (27) să nu admită numai soluția banală x = 0 , trebuie ca :

$\det(A - \lambda \cdot I) = 0$ (28)

Condiția (28) ne dă valorile lui λ , care sunt rădăcinile polinomului caracteristic .

Teorema 1 : Matricile A și B ale unui operator liniar scris în două baze au același polinom caracteristic (deci și aceleași valori proprii) .

$B = S^{-1} A S$

$\det(B -\lambda \cdot I) = \det(S^{-1} A S - \lambda \cdot I) =$

$= \det(S^{-1} (A - \lambda \cdot I) S) =$

$= \det(S^{-1}) \cdot \det(A - \lambda \cdot I) \cdot \det(S) =$

$= \det(A - \lambda \cdot I)$

deoarece :

$\det(S^{-1}) \cdot \det(S) = 1$

Teorema 2 : Proprietatea vectorului x de a fi propriu pentru un operator liniar nu depinde de alegerea bazei .

$A x = \lambda \cdot x$ și $x = S \xi \Rightarrow A S \xi = \lambda \cdot S \xi \Rightarrow$

$\Rightarrow S^{-1} A S \xi = \lambda \cdot \xi \Rightarrow B \xi = \lambda \cdot \xi$

x și ξ reprezintă coordonatele vectorului x în bazele e și ε .

Fie un operator liniar A , căruia îi corespunde o matrice pătratică cu elemente reale ($a_{kj} \in \mathbb{R}$) .

Să studiem produsul scalar :

$$(A x , y) = \sum_{j=1}^{n} (A x)_j \cdot y_j^* = \sum_{j=1}^{n} \sum_{k=1}^{n} a_{jk} \cdot x_k \cdot y_j^* =$$

rebotezăm indicii de sumare

$$= \sum_{j=1}^{n} \sum_{k=1}^{n} a_{kj} \cdot x_j \cdot y_k^* = \left(\sum_{j=1}^{n} \sum_{k=1}^{n} a_{kj} \cdot y_k \cdot x_j^* \right)^* =$$

$$= (A^T y , x)^* = (x , A^T y) \qquad (29)$$

Dacă matricea A este reală și simetrică ($A^T = A$) , vom avea :

$$(A x , y) = (x , A y) \qquad (30)$$

<u>Teorema 3</u> : Toate valorile proprii ale unei matrici simetrice cu elemente reale sunt reale .

Din (30) , pentru x = y , vom avea :

$$(A x , x) = (x , A x) \qquad (31)$$

Folosim (26) și rezultă :

$$(\lambda \cdot x , x) = (x , \lambda \cdot x)$$

$$\lambda \cdot (x , x) = \lambda^* \cdot (x , x)$$

Și deoarece $x \neq 0$, rezultă $\lambda = \lambda^*$, deci $\lambda \in \mathbb{R}$.

<u>Teorema 4</u> : Vectorii proprii ai unei matrici simetrice cu elemente reale , corespunzând unor valori proprii distincte , sunt ortogonali .

Fie :

$$A x = \lambda_1 x$$

$$A y = \lambda_2 y$$

$$(A x , y) = (\lambda_1 \cdot x , y) = \lambda_1 \cdot (x , y) \qquad (32)$$

$$(x , A y) = (x , \lambda_2 \cdot y) = \lambda_2 \cdot (x , y) \qquad (33)$$

Din (30) , (32) și (33) rezultă :

$$(\lambda_1 - \lambda_2) \cdot (x , y) = 0 \quad \Rightarrow \quad (x , y) = 0$$

In cazul a doi vectori ce corespund unei aceleiași valori proprii , ai unei matrici simetrice cu elemente reale , ei sunt ori proporționali , ori liniar independenți , și în acest din urmă caz ei pot fi ortogonalizați prin procedeul Gram-Schmidt .

In concluzie , dacă alegem ca bază în spațiul vectorial chiar mulțimea vectorilor proprii , ortogonalizați și normați , ai unui operator liniar corespunzând unei matrici simetrice cu elemente reale , matricea operatorului în această bază va fi diagonală , iar pe diagonala ei vor fi chiar valorile proprii .

<u>6) Metoda Jacobi pentru determinarea vectorilor si valorilor proprii ai unei matrici simetrice reale</u>

Scopul urmărit este ca să aducem operatorul A , prin schimbări succesive ale bazei , la forma diagonală .

$$B = \begin{pmatrix} \lambda_1 & 0 & .. & 0 \\ 0 & \lambda_2 & .. & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & .. & \lambda_n \end{pmatrix} \qquad (34)$$

In această formă elementele de pe diagonală sunt chiar valorile proprii , vectorii proprii având componentele :

$$\xi^1 = \begin{pmatrix} 1 \\ 0 \\ . \\ 0 \end{pmatrix} \quad , \quad \xi^2 = \begin{pmatrix} 0 \\ 1 \\ . \\ 0 \end{pmatrix} \quad , \ldots , \quad \xi^n = \begin{pmatrix} 0 \\ 0 \\ . \\ 1 \end{pmatrix} \qquad (35)$$

Trebuie să determinăm matricea S a schimbării de bază , vectorii proprii căutați având matricile coloană ale componentelor în baza inițială date de :

$$x^i = S\ \xi^i \qquad (35')$$

Aducerea matricii la forma diagonală se face printr-un procedeu iterativ . Urmărim ca , după fiecare iterație , să scadă suma pătratelor elementelor nediagonale . Când această sumă a ajuns neglijabil de mică , procesul iterativ se oprește :

$$d_A = \sum_{\substack{i,j \\ i \neq j}} a_{ij}^2 < eps \Rightarrow \text{oprirea iterației} \qquad (36)$$

Fiecare iterație va consta dintr-o schimbare de bază de forma :

$$S_{\alpha\beta} = \begin{pmatrix} 1 & & & & 0 \\ & \cdot & & \cdot & \\ & \cos\theta & & -\sin\theta & \\ & & \cdot 1 \cdot & & \\ & \sin\theta & & \cos\theta & \\ 0 & \cdot & & & \cdot 1 \end{pmatrix} \begin{matrix} \\ \\ \text{linia } \alpha \\ \\ \text{linia } \beta \\ \\ \end{matrix} \qquad (37)$$

$\alpha \qquad \beta \qquad$ (coloanele)

Această matrice se aseamănă foarte mult cu matricea unitate , singurele diferențe fiind la elementele situate la intersecția liniilor α și β cu coloanele α și β .

Se poate verifica ușor că $S_{\alpha\beta}$ este o matrice ortogonală .

$$S_{\alpha\beta}^{-1} = S_{\alpha\beta}^T = \begin{pmatrix} 1 & & & & 0 \\ & \cdot & & \cdot & \\ & \cos\theta & & \sin\theta & \\ & & \cdot 1 \cdot & & \\ & -\sin\theta & & \cos\theta & \\ 0 & \cdot & & & \cdot 1 \end{pmatrix} \qquad (38)$$

Matricea $S_{\alpha\beta}^T$ aplicată la stânga matricii A va afecta liniile α și β :

$L_\alpha \Rightarrow \cos\theta \cdot L_\alpha + \sin\theta \cdot L_\beta$ (*)

$L_\beta \Rightarrow -\sin\theta \cdot L_\alpha + \cos\theta \cdot L_\beta$

Matricea $S_{\alpha\beta}$ aplicată la dreapta matricii A va afecta coloanele α și β :

$C_\alpha \Rightarrow \cos\theta \cdot C_\alpha + \sin\theta \cdot C_\beta$ (**)

$C_\beta \Rightarrow -\sin\theta \cdot C_\alpha + \cos\theta \cdot C_\beta$

Astfel , vom avea patru categorii de elemente ale matricii B , după cum se transformă la schimbarea de bază .

$$B = \begin{pmatrix} 1 & & & & 0 \\ & \cos\theta & & \sin\theta & \\ & & 1 & & \\ & -\sin\theta & & \cos\theta & \\ 0 & & & & 1 \end{pmatrix} A \begin{pmatrix} 1 & & & & 0 \\ & \cos\theta & & -\sin\theta & \\ & & 1 & & \\ & \sin\theta & & \cos\theta & \\ 0 & & & & 1 \end{pmatrix} \quad (39)$$

Elementele ce nu se află pe liniile sau coloanele α și β rămân nemodificate de schimbarea de bază :

$$b_{ij} = a_{ij} \quad \text{pentru } i,j \neq \alpha,\beta \quad (40)$$

Elementele ce se află pe liniile α sau β , dar nu și pe coloanele α sau β , se vor transforma comform (*) în :

$$b_{\alpha j} = \cos\theta \cdot a_{\alpha j} + \sin\theta \cdot a_{\beta j} \quad \text{pentru } j \neq \alpha,\beta \quad (41)$$
$$b_{\beta j} = -\sin\theta \cdot a_{\alpha j} + \cos\theta \cdot a_{\beta j} \quad \text{pentru } j \neq \alpha,\beta \quad (42)$$

Elementele ce se află pe coloanele α sau β , dar nu și pe liniile α sau β , se vor transforma comform (**) în :

$$b_{j\alpha} = \cos\theta \cdot a_{j\alpha} + \sin\theta \cdot a_{j\beta} \quad \text{pentru } j \neq \alpha,\beta \quad (43)$$
$$b_{j\beta} = -\sin\theta \cdot a_{j\alpha} + \cos\theta \cdot a_{j\beta} \quad \text{pentru } j \neq \alpha,\beta \quad (44)$$

Acestea în practică nu vor mai fi calculate , ținând cont că ele pot fi scoase din simetria matricii B .

Proprietatea matricii unui operator liniar de a fi simetrică nu este afectată de o schimbare de bază (bazele sunt ortonormate) :

$$A^T = A \quad \Rightarrow \quad B^T = (S^T A S)^T = S^T A S = B$$

Deci , după cum se și observă din (41-44) , avem :

$$b_{j\alpha} = b_{\alpha j}$$
$$b_{j\beta} = b_{\beta j}$$

Ultima categorie de elemente este formată din cele patru elemente ce se află la intersecția liniilor α și β cu coloanele α și β . In urma acțiunii matricii $S_{\alpha\beta}{}^T$ la stânga , elementele se vor transforma comform cu ecuațiile (41-42) , dar în urma acțiunii și a matricei $S_{\alpha\beta}$ la dreapta , elementele vor suferi o transformare suplimentară de tipul celei date de ecuațiile (43-44) , în care elementele din dreapta egalului (adică elementele a_{xy} din ecuațiile (40-41)) vor fi de fapt elementele rezultate din prima transformare (adica elementele b_{xy} din ecuațiile (41-42)) . Astfel :

$$b_{\alpha\alpha} = \cos\theta \cdot (\cos\theta \cdot a_{\alpha\alpha} + \sin\theta \cdot a_{\beta\alpha}) +$$
$$+ \sin\theta \cdot (\cos\theta \cdot a_{\alpha\beta} + \sin\theta \cdot a_{\beta\beta}) =$$
$$= \cos\theta^2 \cdot a_{\alpha\alpha} + \sin\theta^2 \cdot a_{\beta\beta} + \sin(2\theta) \cdot a_{\alpha\beta} \quad (45)$$

$$b_{\beta\beta} = -\sin\theta \cdot (-\sin\theta \cdot a_{\alpha\alpha} + \cos\theta \cdot a_{\beta\alpha}) +$$
$$+ \cos\theta \cdot (-\sin\theta \cdot a_{\alpha\beta} + \cos\theta \cdot a_{\beta\beta}) =$$
$$= \cos\theta^2 \cdot a_{\beta\beta} + \sin\theta^2 \cdot a_{\alpha\alpha} - \sin(2\theta) \cdot a_{\alpha\beta} \quad (46)$$

$$b_{\alpha\beta} = -\sin\theta\cdot(\cos\theta\cdot a_{\alpha\alpha} + \sin\theta\cdot a_{\beta\alpha}) + \cos\theta\cdot(\cos\theta\cdot a_{\alpha\beta} + \sin\theta\cdot a_{\beta\beta}) = \frac{\sin(2\theta)}{2}\cdot(a_{\beta\beta} - a_{\alpha\alpha}) + \cos(2\theta)\cdot a_{\alpha\beta} \quad (47)$$

Datorită simetriei matricii B vom avea :

$b_{\beta\alpha} = b_{\alpha\beta}$

Să calculăm acum suma pătratelor elementelor diagonale ale matricii B , și să găsim o condiție pentru θ astfel încât $d_B<d_A$, adică transformarea să ne apropie de forma diagonală a matricii .

$$d_B = \sum_{\substack{i,j\\ i\neq j}} b_{ij}^2 = \sum_{\substack{i,j\\ i\neq j\\ i,j\neq\alpha,\beta}} a_{ij}^2 + \sum_{\substack{j\\ j\neq\alpha,\beta}} 2\cdot\left\{ \cos\theta^2\cdot a_{\alpha j}^2 + \sin\theta^2\cdot a_{\beta j}^2 + \sin(2\theta)\cdot a_{\alpha j}\cdot a_{\beta j} \right\} +$$

$$+ \sum_{\substack{j\\ j\neq\alpha,\beta}} 2\cdot\left\{ \sin\theta^2\cdot a_{\alpha j}^2 + \cos\theta^2\cdot a_{\beta j}^2 - \sin(2\theta)\cdot a_{\alpha j}\cdot a_{\beta j} \right\} + 2\cdot b_{\alpha\beta}^2 =$$

$$= \sum_{\substack{i,j\\ i\neq j\\ i,j\neq\alpha,\beta}} a_{ij}^2 + \sum_{\substack{j\\ j\neq\alpha,\beta}} 2\cdot\left[a_{\alpha j}^2 + a_{\beta j}^2\right] + 2\cdot b_{\alpha\beta}^2 =$$

$$= \sum_{\substack{i,j\\ i\neq j}} a_{ij}^2 + 2\cdot(b_{\alpha\beta}^2 - a_{\alpha\beta}^2) =$$

$$= d_A + 2\cdot(b_{\alpha\beta}^2 - a_{\alpha\beta}^2) \quad (48)$$

Observăm că dacă impunem condiția ca :

$$b_{\alpha\beta} = 0 \quad (49)$$

va rezulta automat $d_B<d_A$, iar diferența va fi cu atât mai mare , cu cât $|a_{\alpha\beta}|$ este mai mare .

In practică vom căuta indicii α și β pentru care modulul elementului matricii A este maxim . Apoi , pentru acești indici , vom pune condiția (49) , pentru a-l găsi pe θ .

$$b_{\alpha\beta} = \frac{\sin(2\theta)}{2}\cdot(a_{\beta\beta} - a_{\alpha\alpha}) + \cos(2\theta)\cdot a_{\alpha\beta} = 0$$

Vom avea următoarele cazuri :

1) $a_{\beta\beta} = a_{\alpha\alpha}$

$$\cos(2\theta) = 0 \quad \Rightarrow \quad 2\theta = \frac{\pi}{2} \quad \Rightarrow \quad \theta = \frac{\pi}{4}$$

2) $a_{\beta\beta} \neq a_{\alpha\alpha}$

$$tg(2\theta) = \frac{2 \cdot a_{\alpha\beta}}{a_{\alpha\alpha} - a_{\beta\beta}} \quad \Rightarrow \quad \theta = \frac{1}{2} \cdot arctg\left(\frac{2 \cdot a_{\alpha\beta}}{a_{\alpha\alpha} - a_{\beta\beta}}\right)$$

Cu valoarea lui θ astfel aflată se va calcula matricea rezultată în urma schimbării de bază . Se vor efectua atâtea transformări , până când condiția (36) va fi indeplinită . Valorile proprii căutate sunt chiar elementele de pe diagonala matricii finale .

7) Programul PASCAL

Iată în continuare un program ce calculează prin metoda Jacobi valorile proprii ale unei matrice simetrice cu elemente reale :

```
PROGRAM JACOBI;
const eps=1E-10;
var A,B:array[1..10,1..10] of real;
    n,i,j,alfa,beta:1..10;
    d,max,teta:real;

begin
{ CITIREA MATRICII CAREIA II CAUTAM VAL PR }
write('Ordinul matricii (1..10) :  n = ');
readln(n);
writeln('Introduceti elementele matricii');
for i:=1 to n do
for j:=1 to i do begin
   write('A(',i,',',j,') = '); readln(A[i,j]);
   A[j,i]:=A[i,j]; end;
{ SE DESCHIDE CICLUL ITERATIILOR }
repeat
   { GASIREA INDICILOR ELEM DE MODUL MAXIM }
   beta:=2 ; alfa:=1; max:=abs(A[2,1]);
   for i:=2 to n do
   for j:=1 to i-1 do
     if abs(A[i,j])>max
     then begin beta:=i; alfa:=j;
          max:=abs(A[i,j]); end;
   { DETERMINAREA UNGHIULUI TETA }
   if A[beta,beta]=A[alfa,alfa]
   then teta:=pi/4
   else teta:=1/2*arctan(2*A[alfa,beta]/
                    (A[alfa,alfa]-A[beta,beta]));
   { CALCULAREA ELEM MATRICII TRANSFORMATE B }
   B:=A;
   for j:=1 to n do
   if (j<>alfa) and (j<>beta) then begin
     B[alfa,j]:=cos(teta)*A[alfa,j] +
                sin(teta)*A[beta,j];
```

```
      B[beta,j]:=-sin(teta)*A[alfa,j] +
                  cos(teta)*A[beta,j];
      B[j,alfa]:=B[alfa,j];
      B[j,beta]:=B[beta,j]; end;
   B[alfa,alfa]:=sqr(cos(teta))*A[alfa,alfa] +
                  sqr(sin(teta))*A[beta,beta] +
                  sin(2*teta)*A[alfa,beta];
   B[beta,beta]:=sqr(cos(teta))*A[beta,beta] +
                  sqr(sin(teta))*A[alfa,alfa] -
                  sin(2*teta)*A[alfa,beta];
   B[alfa,beta]:=sin(2*teta)/2*
                 ( A[beta,beta] - A[alfa,alfa] ) +
                  cos(2*teta)*A[alfa,beta];
   B[beta,alfa]:=B[alfa,beta];
   { INLOCUIREA VECHII MATRICI CU CEA NOUA }
   A:=B;
   { VERIFICAREA CONDITIEI DE IESIRE DIN CICLU }
   d:=0;
   for i:=2 to n do
   for j:=1 to i-1 do d:=d+sqr(A[i,j]);
   d:=2*d; { si elem de deasupra diagonalei }
until d<eps;
{ SCRIEREA VALORILOR PROPRII }
writeln('Valorile proprii sunt');
for i:=1 to n do writeln(i,'     ',A[i,i]);
readln;
end.
```

8) Probleme

1) Pentru mai multe matrici simetrice cu elemente reale , calculați valorile proprii prin metoda Jacobi , apoi pentru comparație și prin metodele coeficienților nedeterminați și Leverrier . Metoda Jacobi se folosește cu precădere atunci când matricea are o dimensiune foarte mare , ce face greoaie sau chiar imposibilă rezolvarea de sisteme liniare cu această dimensiune

2) Pornind de la programul prezentat , realizați un program care să calculeze și vectorii proprii , comform formulei (35') .

CAP XV. METODA MONTE-CARLO

1) Numere aleatorii

In multe procese din fizică rezultatele pot fi prevăzute doar statistic , pentru un singur experiment existând mai multe posibilități de stări finale , fiecare dintre aceste posibilități fiind descrisă de probabilitatea de realizare . In cazul în care parametrul ce face diferența între stările finale are o variație discretă vorbim de probabilitate , iar în cazul în care are o variație continuă vorbim de densitate de probabilitate .

Exemplul 1 : Aruncarea zarului
Parametrul ce deosebește între ele stările finale este valoarea zarului , deci are o variație discretă . Să notăm cu i (i=1..6) cele 6 stări finale posibile . Fiecare stare va avea probabilitatea de realizare P_i . Probabilitatea de realizare a unui eveniment oarecare este 1 (zarul trebuie obligatoriu să ia una dintre cele 6 valori) , deci , adunând probabilitățile evenimentelor incompatibile , obținem :

$$\sum_{i=1}^{6} P_i = 1 \tag{1}$$

In cazul în care zarul nu este trucat , aceste probabilități sunt egale toate cu 1/6 .

Exemplul 2 : Trasul cu arcul la țintă
Parametrul ce deosebește între ele stările finale este distanța până la centrul țintei , deci are o variație continuă . Să notăm cu r stările finale posibile ($r \in [0,\infty)$) . Probabilitatea ca săgeata să se afle în intervalul [r,r+dr] va fi dată de $P(r)\cdot dr$, unde dr este o variație foarte mică a lui r , iar P(r) este densitatea de probabilitate . Adunând probabilitățile tuturor evenimentelor finale , care sunt și evenimente incompatibile , obținem :

$$\int_0^{\infty} P(r)\cdot dr = 1 \tag{2}$$

Dacă presupunem că arcașul este suficient de bun pentru a nu trimite nici o sageată la o distanță de centrul țintei mai mare de R , și că în intervalul [0,R] densitatea de probabilitate nu depinde de r , vom obține :

$$\int_0^R P(r)\cdot dr = P\cdot\int_0^R dr = P\cdot R = 1 \quad \Rightarrow \quad P(r) = P = \frac{1}{R}$$

Simularea pe calculator a acestor procese presupune în primul caz generarea aleatorie a unui număr întreg între 1 și 6 , cu probabilități egale , și în cel de-al doilea caz generarea aleatorie a unui număr între 0 și R , cu densitate de probabilitate constantă .

2) Numere pseudoaleatorii

Deoarece orice program pe calculator pleacă de la niște date de intrare , le prelucrează , și generează niște date de ieșire ce sunt univoc determinate de datele de intrare , este imposibilă realizarea unor programe care să genereze numere aleatorii , firul cauzalității neputând fi rupt între intrarea și ieșirea din program .

O soluție costisitoare și incomodă pentru obținerea unui generator de numere aleatorii este construirea unui aparat care să transforme zgomotul termic dintr-un dispozitiv electronic în numere . O deficiență a acestei metode este și faptul că nu putem controla cu precizie densitatea de probabilitate ce caracterizează aparatul .

In majoritatea cazurilor însă este suficient de bine să lucrăm nu cu numere aleatorii , ci cu numere pseudoaleatorii . Acestea provin dintr-un număr inițial și un algoritm care transformă un număr în alt număr . In felul acesta se generează un șir de numere , perfect determinate de numărul inițial , care aproximează suficient de bine un șir de numere aleatorii .

Programul PASCAL are un astfel de generator de numere pseudoaleatorii . Stabiliea numărului inițial se face cu comanda RANDOMIZE , care se folosește de un contor de ceas intern al calculatorului , generând la diferiți timpi diferite numere . Periodicitatea contorului limitează la un număr finit plaja de numere inițiale generate prin RANDOMIZE . Odată stabilit numărul inițial , numerele pseudoaleatorii se vor obține cu comanda RANDOM(range:word):type , și vor fi cuprinse între 0 inclusiv și range exclusiv , având tipul variabilei căreia i se atribuie numărul . In cazul în care nu se specifică parametrul range , acesta este automat luat 1 . In cazul discret numerele vor avea probabilități de apariție egale , iar în cazul continuu numerele vor avea toate aceeași densitate de probabilitate de apariție (acest lucru îl asigură algoritmul folosit de RANDOM) .

Vom spune că numerele sunt uniform distribuite .

Pentru simularea primului exemplu vom avea :

```
var i:integer;
randomize;
i:=1+random(6);
```

Pentru cel de-al doilea exemplu vom avea :

```
var r:real;
randomize;
r:=random(R);
```

In continuare prin numere aleatorii generate de calculator vom subânțelege numere care de fapt sunt pseudoaleatorii .

3) Metoda Monte-Carlo in calculul unor integrale (suprafete)

Fie $F=\int_a^b f(x)\cdot dx$, cu $0\le f(x)\le M$.

Să construim dreptunghiul de laturi (b-a) și M , și să generăm în acest dreptunghi puncte aleatorii , de coordonate $x\in[a,b]$ și $y\in[0,M]$.

Metoda Monte-Carlo are la bază faptul că pentru un număr suficient de mare de puncte aleatorii uniform distribuite , raportul numerelor de puncte din două suprafețe tinde spre raportul acestor suprafețe .

Pentru cazul nostru coordonatele unui punct aleatoriu vor fi date de :

```
x:=a+random(b-a);
y:=random(M);
```

Numărul de puncte (n) din dreptunghi va fi tocmai numărul total de puncte , iar numărul de puncte (m) pentru care $f(x)\le y$ va fi numărul de puncte din suprafața delimitată între graficul funcției f(x) și axa Ox . Suprafața din urmă este tocmai integrala F .

Suprafața dreptunghiului este $(b-a)\cdot M$.

Deci știind raportul de puncte și suprafața dreptunghiului , putem afla integrala F .

$$F = \frac{m}{n}\cdot(b-a)\cdot M \qquad (3)$$

4) Metoda Monte-Carlo in calculul unor volume

Metoda se generalizează pentru un spațiu n-dimensional , unde raportul de numere de puncte din două volume va tinde spre raportul de volume. Să demonstrăm această afirmație . Coordonatele fiecărui punct vor fi date de n numere aleatorii uniform generate .

Fie deci $P_i\cdot dx_i$ probabilitatea ca $x_i\in[x_i,x_i+dx_i]$, cu i=1..n .

Atunci probabilitatea ca punctul aleator $(x_1,x_2,..,x_n)$ să aparțină elementului de volum

$(x_1,x_1+dx_1)\otimes(x_2,x_2+dx_2)\otimes..\otimes(x_n,x_n+dx_n)$ este :

$P = P_1\cdot dx_1\cdot P_2\cdot dx_2\cdot..\cdot P_n\cdot dx_n =$
$= P_1\cdot P_2\cdot..\cdot P_n\cdot dx_1\cdot dx_2\cdot..\cdot dx_n$

Probabilitatea ca punctul aleator să aparţină unui volum Ω este :

$$P(\Omega) = \int_{(\Omega)} P_1\cdot P_2\cdot..\cdot P_n\cdot dx_1\cdot dx_2\cdot..\cdot dx_n =$$

$$= P_1\cdot P_2\cdot..\cdot P_n\cdot \int_{(\Omega)} dx_1\cdot dx_2\cdot..\cdot dx_n =$$

$$= P_1\cdot P_2\cdot..\cdot P_n\cdot\Omega \qquad (4)$$

unde am considerat volumul $\Omega = \int_{(\Omega)} dx_1\cdot dx_2\cdot..\cdot dx_n$

Fie Ω_1 şi Ω_2 două volume , şi N puncte aleatorii . Dintre acestea N_1 aparţin lui Ω_1 şi N_2 aparţin lui Ω_2 . Pentru un N foarte mare avem

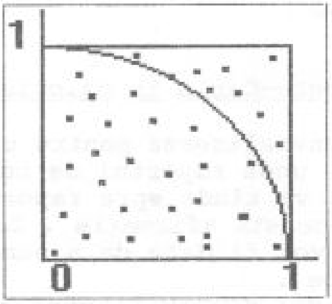
$$\frac{N_1}{N_2} = \frac{\frac{N_1}{N}}{\frac{N_2}{N}} \approx \frac{P(\Omega_1)}{P(\Omega_2)} = \frac{P_1\cdot P_2\cdot..\cdot P_n\cdot\Omega_1}{P_1\cdot P_2\cdot..\cdot P_n\cdot\Omega_2} = \frac{\Omega_1}{\Omega_2} \qquad (5)$$

Am ţinut cont de faptul că pentru $N \Rightarrow \infty$ avem :

$\frac{N_1}{N} \Rightarrow P(\Omega_1) \qquad \frac{N_2}{N} \Rightarrow P(\Omega_2)$

<u>5) Programul PASCAL</u>

Programul este o aplicaţie a metodei Monte-Carlo pentru următorul caz : punctele sunt uniform generate în interiorul unui pătrat de latură 1 şi cu colţul din stânga jos în origine , suprafaţa a cărei arie vrem să o determinăm fiind un sfert de cerc cu centrul în origine şi cu raza 1 .



Punctele vor fi generate aleator în modul următor :

```
x:=random;
y:=random;
```

Punctele ce aparțin sfertului de cerc (m) vor fi date de :

$x^2 + y^2 \le 1$

Punctele ce aparțin pătratului vor fi toate punctele (n) .

Raportul de puncte va fi aproximativ raportul de suprafețe :

$$\frac{m}{n} \approx \frac{\pi/4}{1} \cdot \Rightarrow \quad \pi \approx \frac{4 \cdot m}{n}$$

In felul acesta este determinat π , cu o precizie ce este cu atât mai bună cu cât numărul total de puncte aleatorii n este mai mare .

Iată în continuare programul , care și reprezintă grafic punctele :

```
{$F+}
PROGRAM PI;
uses graphics,graph,crt;
var n,m,i:longint; x,y,p:real; ps:string;

FUNCTION SFERTCERC(x:real):real;
begin sfertcerc:=sqrt(1-sqr(x)); end;

begin
writeln('Det lui pi prin metoda Monte Carlo');
write('nr (longint) de pct aleat este     n = ');
readln(n);
randomize;
init_graph_mode; YperX:=10;
init_window(0,maxy-20,0,maxy-20,0,1,0,1);
graph_functie(sfertcerc,red);
m:=0;
for i:=1 to n do begin
   x:=random; y:=random;
   putpixel(xe(x),ye(y),white);
   if sqr(x)+sqr(y)<1 then m:=m+1; end;
p:=4*m/n; str(p,ps);
outtextxy(0,maxy-10,concat('pi = ',ps));
repeat until readkey='q'; closegraph;
end.
```

6) Problema

Alcătuiți un program care pe baza metodei Monte-Carlo , pentru fiecare din următoarele valori ale numărului total de puncte n : 100 , 400 , 1600 , să îl aproximeze pe π de 10000 ori . Cu valorile astfel obținute , pentru fiecare n în parte să se calculeze valoarea medie a lui π , și abaterea pătratică medie . Să se pună în evidență că cele 10000 de valori sunt distribuite în jurul valorii medii cu o densitate de forma clopotului

lui Gauss . Să se arate că legătura între abaterea pătratică medie și numărul total de puncte este de forma $\sigma \sim \sqrt{n}$.

Răspuns :

```
PROGRAM PIEROR;
uses graphics,graph,crt;
var p:array[1..10000] of real;           {punctele}
    d:array[1..30] of integer;         {densitatea}
    i,k:integer; n:longint;
    pimin,pimax,h,dmax,m,s:real;

FUNCTION GETPI(n:longint):real;
var m,i:longint; x,y:real;
begin
m:=0;
for i:=1 to n do begin
   x:=random; y:=random;
   if sqr(x)+sqr(y)<1 then m:=m+1; end;
getpi:=4*m/n;
end;

begin
writeln('introduceti numarul de pct aleatoare');
write('n = '); readln(n);
{stabilirea valorilor pentru pi}
randomize;
for i:=1 to 10000 do p[i]:=getpi(n);
{stabilirea valorii medii}
m:=0;
for i:=1 to 10000 do m:=m+p[i];
m:=m/10000; writeln('m = ',m);
{stabilirea abaterii standard}
s:=0;
for i:=1 to 10000 do s:=s+sqr(p[i]-m);
s:=sqrt(s/10000); writeln('s = ',s);
{stabilirea lim intre care se gasesc val pt pi}
pimin:=p[1]; pimax:=p[1];
for i:=1 to 10000 do begin
   if p[i]>pimax then pimax:=p[i];
   if p[i]<pimin then pimin:=p[i]; end;
h:=pimax-pimin;
pimin:=pimin-h; pimax:=pimax+h; h:=3*h;
{stabilirea densitatii de puncte}
for k:=1 to 30 do d[k]:=0;
for i:=1 to 10000 do begin
   k:=trunc((p[i]-pimin)/h*30)+1;
   d[k]:=d[k]+1; end;
{stabilirea densitatii maxime}
dmax:=d[1];
for k:=1 to 30 do if d[k]>dmax then dmax:=d[k];
dmax:=dmax*1.1;
{reprezentarea grafica}
writeln('apasati o tasta pentru grafic');
repeat until keypressed;
```

```
init_graph_mode;
init_window(0,maxx,0,maxy,1,30,0,dmax);
histo:=1;
for k:=1 to 30 do punct_histo(k,d[k]);
repeat until readkey='q'; closegraph;
end.
```

S-au obținut urmăroarele rezultate :

$n = 100 \qquad \sigma = 1.63\cdot10^{-1}$

$n = 400 \qquad \sigma = 8.27\cdot10^{-2}$

$n = 1600 \qquad \sigma = 4.15\cdot10^{-2}$

CAP XVI. CONVOLUTIA SI DECONVOLUTIA SPECTRELOR

1) Dezvoltarea in serie Fourier

O funcție f(x) , continuă și periodică de perioadă 2π , se descompune în serie Fourier astfel :

$$f(x) = b_0 + \sum_{n=1}^{\infty} a_n \cdot \sin(n \cdot x) + \sum_{n=1}^{\infty} b_n \cdot \cos(n \cdot x)$$

unde coeficienții b_0 , a_n și b_n sunt definiți astfel :

$$b_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x) \cdot dx \qquad b_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \cdot \cos(n \cdot x) \cdot dx$$

$$a_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \cdot \sin(n \cdot x) \cdot dx$$

Extinderea intervalului de periodicitate de la 2π la perioada T pentru cazul general de funcție periodică se face astfel :

$$f(x) = b_0 + \sum_{n=1}^{\infty} a_n \cdot \sin\left(\frac{2\pi n x}{T}\right) + \sum_{n=1}^{\infty} b_n \cdot \cos\left(\frac{2\pi n x}{T}\right)$$

unde coeficienții b_0 , a_n și b_n sunt definiți astfel :

$$b_0 = \frac{1}{T} \int_0^{T} f(x) \cdot dx \qquad b_n = \frac{2}{T} \int_0^{T} f(x) \cdot \cos\left(\frac{2\pi n x}{T}\right) \cdot dx$$

$$a_n = \frac{2}{T} \int_0^{T} f(x) \cdot \sin\left(\frac{2\pi n x}{T}\right) \cdot dx$$

In cazul în care funcția este definită doar pe un interval [a,b] astfel încât f(a)=f(b) , putem prelungi prin periodicitate funcția la $(-\infty,\infty)$, având perioada T=b-a .

2) Convolutia spectrelor

Să ne imaginăm următorul experiment : avem o sursă de lumină care emite doar în intervalul de frecvențe [f1,f2] , iar noi dorim să măsurăm numărul de fotoni pe unitatea de frecvență în funcție de frecvență (să obținem spectrul R(f)) . Pentru aceasta împărțim domeniul de frecvențe în m intervale suficient de mici $[f,f+\Delta f]$, iar numărul de fotoni $R(i) \cdot \Delta f$ din aceste intervale îl putem măsura cu un analizor multicanal . Numărul total de fotoni va fi :

$$N = \sum_{i=1}^{m} R(i)\cdot\Delta f \qquad (1)$$

Pentru un anume i , $R(i)\cdot\Delta f$ va fi numărul de fotoni cuprins în intervalul de frecvențe $[f1+(i-1)\cdot\Delta f, f1+i\cdot\Delta f]$.

In limita $\Delta f \Rightarrow 0$, $m \Rightarrow \infty$ astfel încât $f2-f1=m\cdot\Delta f$ vom obține numărul de fotoni pe unitatea de frecvență corespunzător frecvenței din mijlocul intervalului Δf , adică

$R(i) \Rightarrow R(f1+(i-\frac{1}{2})\cdot\Delta f)$, și relația (1) devine :

$$N = \int_{f1}^{f2} R(f)\cdot df \qquad (1')$$

Acest spectru , care este cel real , nu se poate însă obține în practică . Unii fotoni , în urma trierii efectuate de aparat , în loc să apară în intervalul Δf ce le corespunde , vor fi înregistrați un pic mai la stânga sau mai la dreapta , dând contribuții în intervalele Δf vecine . Acest fenomen poate apare atât datorită limitelor aparatului de măsură , cât și imposibilității teoretice de a construi un astfel de aparat ideal : în lumea fotonilor acționează principiul de nedeterminare al lui Heisemberg , fotonii pot tunela bariere de potențial spre zone interzise energetic (trierea efectuată de aparat se face tocmai pe criterii energetice : $E = h\cdot f$). Prin urmare noi vom măsura experimental în intervalul $[f, f+\Delta f]$ un număr de fotoni $E(i)\cdot\Delta f$ diferit de cel real , dar care provine din spectrul real astfel , prin includerea unor fotoni veniți din g intervale vecine din stânga și altele g din dreapta , precum și din înregistrarea în altă parte a unor fotoni din intervalul cercetat :

$$E(i) = \sum_{j=-g}^{g} A(j)\cdot R(i-j)\cdot\Delta f \qquad (2)$$

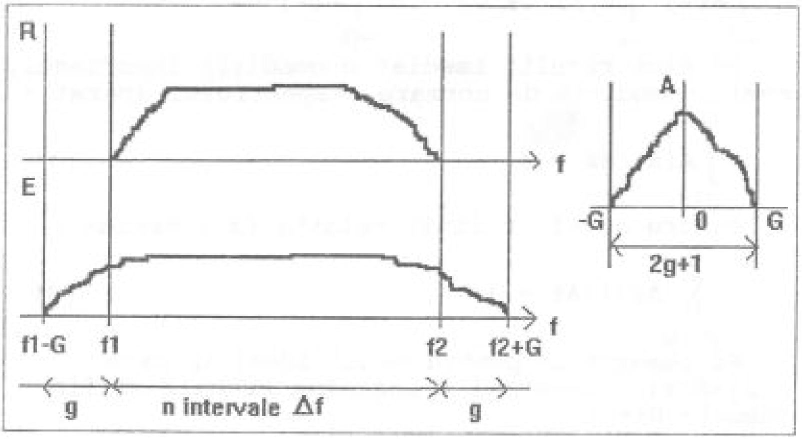
In limita $\Delta f \Rightarrow 0$, $g \Rightarrow \infty$ astfel încât $2G=(2g+1)\cdot\Delta f$ relația (2) duce la :

$$E(f) = \int_{-G}^{G} A(x)\cdot R(f-x)\cdot dx \qquad (2')$$

Operația matematică efectuată este convoluția funcțiilor .

Am considerat că măsurătoarea pt $R(f)$ este afectată de fotonii din intervalul $[f-G, f+G]$, dând astfel pe $E(f)$. $A(x)$ este o funcție specifică aparatului de măsură , definită pe intervalul $[-G, G]$,numită ' spectrul aparatului '

, şi pe care o presupunem ştiută .



Deşi spectrul experimental este diferit de cel real , totuşi numerele totale de fotoni din ambele spectre trebuie să fie egale , aparatul doar clasează unii fotoni în intervale de frecvenţă Δf ce nu le aparţin , neputând creea sau distruge fotoni . Această observaţie ne duce la concluzia :

$$\int_{f1}^{f2} R(f)\cdot df = \int_{f1-G}^{f2+G} E(f)\cdot df = N$$

Fiindcă în afara intervalelor [f1,f2] şi respectiv [f1-G,f2+G] nu mai avem fotoni , putem extinde integralele la (-∞,+∞) :

$$N= \int_{-\infty}^{\infty} R(f)\cdot df = \int_{-\infty}^{\infty} E(f)\cdot df =$$

$$= \int_{-\infty}^{\infty} \int_{-G}^{G} A(x)\cdot R(f-x)\cdot dx\cdot df =$$

schimbăm ordinea de integrare , domeniul de integrare fiind o bandă infinită , centrată simetric pe axa f , şi care intersectează axa x în -G şi G .

$$= \int_{-G}^{G} \int_{-\infty}^{\infty} A(x)\cdot R(f-x)\cdot df\cdot dx =$$

$$= \int_{-G}^{G} A(x) \int_{-\infty}^{\infty} R(f-x)\cdot df\cdot dx =$$

schimbare de variabilă : $f-x=y \Rightarrow x=f-y \Rightarrow dx=df$

$$= \int_{-G}^{G} A(x) \int_{-\infty}^{\infty} R(y)\cdot dy\cdot dx = N \int_{-G}^{G} A(x)\cdot dx$$

De aici rezultă imediat o condiție importantă, numită condiția de normare a spectrului aparat :

$$\int_{-G}^{G} A(x)\cdot dx = 1 \qquad (3')$$

Pentru cazul Δf finit relația (3') devine :

$$\sum_{j=-g}^{g} A(j)\cdot \Delta f = 1 \qquad (3)$$

Se observă că pentru cazul ideal în care $E(x)=R(x)$, spectrul aparatului trebuie să fie o funcție Dirac .

3) Deconvolutia spectrelor

Se pune problema practică de a obține spectrul real pornind de la spectrul experimental și cel al aparatului . Pentru aceasta prelungim prin periodicitate spectrele de la [f1-G,f2+G] la $(-\infty,+\infty)$,și le descompunem apoi în serie Fourier :

$$R(f) = r_0 + \sum_{i=1}^{\infty} \left(r_i\cdot\cos\left(\frac{2\pi if}{L}\right) + \rho_i\cdot\sin\left(\frac{2\pi if}{L}\right) \right) \qquad (4)$$

$$E(f) = e_0 + \sum_{i=1}^{\infty} \left(e_i\cdot\cos\left(\frac{2\pi if}{L}\right) + \varepsilon_i\cdot\sin\left(\frac{2\pi if}{L}\right) \right) \qquad (5)$$

unde r_0 , r_i , ρ_i , e_0 , e_i , ε_i sunt coeficienții Fourier ai dezvoltării în serie , iar L este perioada funcției : $L = f2-f1+2\cdot G$.

Inlocuim funcțiile dezvoltate în serie în ecuația convoluției , ce leagă spectrul experimental de cel real :

$$E(f) = e_0 + \sum_{i=1}^{\infty} \left(e_i\cdot\cos\left(\frac{2\pi if}{L}\right) + \varepsilon_i\cdot\sin\left(\frac{2\pi if}{L}\right) \right) =$$

$$= \int_{-G}^{G} A(x)\cdot R(f-x)\cdot dx =$$

$$= \int_{-G}^{G} A(x)\cdot\left\{ r_0 + \sum_{i=1}^{\infty} \left(r_i\cdot\cos\left(\frac{2\pi i(f-x)}{L}\right) +\right.\right.$$

$$\left.\left. + \rho_i\cdot\sin\left(\frac{2\pi i(f-x)}{L}\right) \right) \right\}\cdot dx =$$

descompunem sinușii și cosinușii de diferențe, și apoi schimbăm între ele integrarea și sumarea

$$= r_0 + \sum_{i=1}^{\infty} \Big\{ r_i \cdot \cos(\frac{2\pi if}{L}) \int_{-G}^{G} A(x) \cdot dx \cdot \cos(\frac{2\pi ix}{L}) +$$

$$+ r_i \cdot \sin(\frac{2\pi if}{L}) \int_{-G}^{G} A(x) \cdot dx \cdot \sin(\frac{2\pi ix}{L}) +$$

$$+ \rho_i \cdot \sin(\frac{2\pi if}{L}) \int_{-G}^{G} A(x) \cdot dx \cdot \cos(\frac{2\pi ix}{L}) -$$

$$- \rho_i \cdot \cos(\frac{2\pi if}{L}) \int_{-G}^{G} A(x) \cdot dx \cdot \sin(\frac{2\pi ix}{L}) \Big\}$$

Pe intervalul [f1-G,f2+G] , de lungime L , sistemul de funcții $\sin(\frac{2\pi if}{L})$ și $\cos(\frac{2\pi if}{L})$ formează un sistem complet de funcții ortogonale. De aceea putem identifica coeficienții acestor funcții din partea stângă și din cea dreaptă a egalității , rezultând :

$$e_0 = r_0$$

$$e_i = r_i \cdot \int_{-G}^{G} A(x) \cdot dx \cdot \cos(\frac{2\pi ix}{L}) -$$

$$- \rho_i \cdot \int_{-G}^{G} A(x) \cdot dx \cdot \sin(\frac{2\pi ix}{L})$$

$$\varepsilon_i = \rho_i \cdot \int_{-G}^{G} A(x) \cdot dx \cdot \cos(\frac{2\pi ix}{L}) +$$

$$+ r_i \cdot \int_{-G}^{G} A(x) \cdot dx \cdot \sin(\frac{2\pi ix}{L})$$

efectuăm notațiile :

$$s_i = \int_{-G}^{G} A(x) \cdot dx \cdot \cos(\frac{2\pi ix}{L}) \qquad (6)$$

$$\sigma_i = \int_{-G}^{G} A(x) \cdot dx \cdot \sin(\frac{2\pi ix}{L})$$

sistemul de ecuații devine :

e0 = r0

$ei = ri \cdot si - \rho i \cdot \sigma i$

$\varepsilon i = \rho i \cdot si + ri \cdot \sigma i$

rezolvîndu-l , obținem :

$$r0 = e0 \tag{7}$$

$$ri = \frac{e_i \cdot s_i + \varepsilon_i \cdot \sigma_i}{s_i^2 + \sigma_i^2}$$

$$\rho i = \frac{\varepsilon_i \cdot s_i - e_i \cdot \sigma_i}{s_i^2 + \sigma_i^2}$$

Am aflat coeficienții dezvoltării în serie Fourier a spectrului real, deci am determinat spectrul real .

Observăm că pentru un spectru aparat simetric (funcție pară) , avem $\sigma i = 0$, deci :

$r0 = e0$
$ri = ei/si$
$\rho i = \varepsilon i/si$

4) Programul PASCAL

Calcularea practică a integralelor din formulele menționate se va realiza prin sumare , împărțind domeniul în intervale Δf . Deoarece numai funcțiile continue și mărginite (și evident periodice) se pot descompune în serie Fourier , va trebui ca intervalele Δf să fie suficient de mici , astfel încât variația spectrului să fie neglijabilă , și deși spectrul nostru prelucrat experimental este prin esența lui discret , noi să-l putem aproxima ca fiind continuu . Vom limita astfel pe Δf la valoarea lui minimă permisă de aparatul de măsură . In calculul coeficienților Fourier intervin funcțiile sin și cos . Pentru o precizie bună în calcularea acestor coeficienți trebuie ca sumele noastre să aproximeze bine integralele . Dar funcțiile $\sin(n \cdot x)$ și $\cos(n \cdot x)$, pe măsură ce n crește , au variații tot mai rapide , și integralele sunt din ce în ce mai prost aproximate de sumă . Apare astfel o tendință de limitare a numărului de termeni ai dezvoltării în serie Fourier . Pe de altă parte , cu cît avem mai mulți termeni în dezvoltare , cu atît spectrele noastre vor fi mai precis calculate . Alegând o soluție de compromis între cele două tendințe , numărul de termeni este astfel stabilit . Programul cere această valoare , permițând modificarea ei .

Pentru a calcula practic , prin sumare , integralele in care apar funcțiile sin și cos , argumentul acestor funcții va fi luat la mijlocul intervalului Δf . In felul acesta realizăm o bună aproximație a integralei din produs de funcții .

In urma împărțirii domeniului [-G,G] în 2g+1 intervale Δf , indexate cu i , i=-g..g , valoarea lui x în mijlocul intervalului Δf va fi $x=i \cdot \Delta f$. Rezultă că :

$$\frac{2\pi jx}{L} = \frac{2\pi ji \cdot \Delta f}{L} = \frac{2\pi ji}{n}$$

In urma împărțirii domeniului [0,L] în n intervale Δf , indexate cu i , i=1..n , valoarea lui x în mijlocul intervalului Δf va fi $x=(i+1/2)\cdot\Delta f$. Rezultă că :

$$\frac{2\pi jx}{L} = \frac{2\pi j(i+1/2)\cdot \Delta f}{L} = \frac{2\pi j(i+1/2)}{n}$$

Aceste forme le vom folosi în calculul diferitelor integrale , în care apar funcțiile sin și cos , pe cele două domenii .

Spectrul real va conține inițial m intervale Δf în care să avem valori nenule , spectrul aparatului va conține 2g+1 intervale Δf , iar după convoluție spectrul experimental va crește cu g intervale Δf la stânga și la dreapta , ajungând să aibă n=m+2g intervale Δf în care să aibă valori nenule .

Spectrele de prelucrat (real , experimental și al aparatului) se vor afla în fișiere de forma :

n	nr de puncte ale spectrului experimental (real)
g	nr de puncte din jumătate din spectrul aparatului
punctele	cele n puncte , respectiv (2·g+1) puncte

Pentru a preveni folosirea greșită a fișierelor în convoluții și deconvoluții , programul cercetează mai întîi ca parametrii n și g să fie aceiași .

Punctele spectrelor vor fi stocate în trei matrici , r , e , a , numite astfel după spectrul ce-l conțin : real, experimental, al aparatului .

Pentru deconvoluție , coeficienții Fourier ai spectrului real vor fi stocați in r0 , r1 și r2 .

Din condiția de normare a spectrului aparatului este calculat Δf :

$$\Delta f = \left\{ \sum_{j=-g}^{g} A(j) \right\}^{-1}$$

Cunoscîndu-l pe Δf , aflăm lungimea spectrelor (perioada) :

$L = n\cdot\Delta f$

Calculele sunt apoi făcute cu aceste valori .

In funcție de comanda tastată , programul va realiza convoluția sau deconvoluția . Sintaxa comenzilor este :

```
condec sp_r sp_a sp_e /c      pentru convoluție
condec sp_e sp_a sp_r /d      pentru deconvoluție
```

unde sp_r , sp_a , sp_e sunt numele unor fișiere.

In ambele situații primele două fișiere preexistă , iar cel de-al treilea va fi creat (nu-i dați un nume pe care să-l mai aibă și un alt

fişier , dacă doriţi ca acesta din urmă să supravieţuiască !) .

Ieşirea din program se face apăsând tasta 'q'.

Pentru a creea un spectru real şi un spectru aparat a fost făcut programul makedem.pas , ce urmează listării programului condec.pas . Iată-le în continuare , nu înainte de a vă sugera comenzile de tastat :

```
makedem
condec real.dem aparat.dem e /c
condec e aparat.dem r /d
```

```
PROGRAM condec; {convolutie si deconvolutie}
uses crt,graph,graphics;
var nume1,nume2,nume3:string;
    fis1,fis2,fis3:text;
    r,e:array[1..4000] of real;
    a:array[-500..500] of real;
    n1,n2,n,g1,g2,g,i,j,k:integer;
    su,dx,L:real;
    s1,s2,e1,e2,r0,max:real;
    r1,r2:array[1..100] of real;

FUNCTION s_cos(j:integer):real;
begin
su:=0;
for i:=-g to g do su:=su+a[i]*cos(2*pi*j*i/n);
s_cos:=su*dx;
end;

FUNCTION s_sin(j:integer):real;
begin
su:=0;
for i:=-g to g do su:=su+a[i]*sin(2*pi*j*i/n);
s_sin:=su*dx;
end;

FUNCTION e_cos(j:integer):real;
begin
su:=0;
for i:=1 to n do
   su:=su+e[i]*cos(2*pi*j*(i-1/2)/n);
e_cos:=su*dx*2/L;
end;

FUNCTION e_sin(j:integer):real;
begin
su:=0;
for i:=1 to n do
   su:=su+e[i]*sin(2*pi*j*(i-1/2)/n);
e_sin:=su*dx*2/L;
end;

begin
{INTRODUCEREA DATELOR}
if paramcount<>4 then halt;
```

```
nume1:=paramstr(1); assign(fis1,nume1);
nume2:=paramstr(2); assign(fis2,nume2);
nume3:=paramstr(3); assign(fis3,nume3);
reset(fis1); reset(fis2); rewrite(fis3);
readln(fis1,n1); readln(fis2,n2);
readln(fis1,g1); readln(fis2,g2);
if n1<>n2 then halt else n:=n1;
if g1<>g2 then halt else g:=g1;
if (n>4000) or (g>500) then halt;
writeln(fis3,n); writeln(fis3,g);
i:=0; if paramstr(4)='/c'
      then repeat i:=i+1; readln(fis1,r[i]);
           until eof(fis1)
      else if paramstr(4)='/d'
           then repeat i:=i+1; readln(fis1,e[i]);
                until eof(fis1)
           else halt;
if i<>n then halt; close(fis1);
i:=-g-1;
repeat i:=i+1; readln(fis2,a[i]);
until eof(fis2);
if i<>g then halt; close(fis2);
{NORMAREA SPECTRULUI APARAT}
su:=0; for j:=-g to g do su:=su+a[j];
dx:=1/su; L:=n*dx;
{SELECTAREA OPERATIUNII DE EFECTUAT}
if paramstr(4)='/c' then begin
{CONVOLUTIA}
for i:=1 to n do begin
  su:=0;
  for j:=-g to g do if (i-j>=1) and (i-j<=n)
                    then su:=su+a[j]*r[i-j];
  su:=su*dx; e[i]:=su; writeln(fis3,e[i]); end;
close(fis3);        end
                    else begin
{DECONVOLUTIA SPECTRULUI}
writeln('nr de puncte ale graficului : ',n);
write('nr termeni(1..100) ai seriei Fourier : ');
readln(k);
if (k<1) or (k>100) then halt;
for j:=1 to k do begin
   s1:=s_cos(j); s2:=s_sin(j);
   e1:=e_cos(j); e2:=e_sin(j);
   r1[j]:=(e1*s1+e2*s2)/(s1*s1+s2*s2);
   r2[j]:=(e2*s1-e1*s2)/(s1*s1+s2*s2); end;
r0:=(1/2)*e_cos(0);
for i:=1 to n do begin
   r[i]:=r0;
   for j:=1 to k do
   r[i]:=r[i]+r1[j]*cos(2*pi*j*i/n)
            +r2[j]*sin(2*pi*j*i/n);
   writeln(fis3,r[i]); end;
close(fis3);        end;
{REPREZENTAREA GRAFICA}
max:=0;
for i:=1 to n do begin
  if r[i]>max then max:=r[i];
```

```
  if e[i]>max then max:=e[i]; end;
max:=max*1.1;
init_graph_mode;
init_window(0,maxx,0,maxy,1,n,0,max);
for i:=1 to n do begin
   putpixel(xe(i),ye(r[i]),red);
   { spectrul real = rosu }
   putpixel(xe(i),ye(e[i]),magenta); end;
   { spectrul experimental = mov }
repeat until readkey='q'; closegraph;
end.
```

```
PROGRAM makedem;
var f:array[0..2001] of real;
    s:array[-201..201] of real;
    ft:text; i,j:integer;
begin
     for i:=0 to 200 do f[i]:=0;
     for i:=200 to 500 do f[i]:=i-200;
     for i:=500 to 1500 do f[i]:=300;
     for i:=1500 to 1800 do f[i]:=1800-i;
     for i:=1800 to 2000 do f[i]:=0;
     assign(ft,'real.dem'); rewrite(ft);
     writeln(ft,2001); writeln(ft,201);
     for i:=0 to 2000 do writeln(ft,f[i]);
     close(ft);

     for j:=-201 to 201 do s[j]:=202-abs(j);
     assign(ft,'aparat.dem'); rewrite(ft);
     writeln(ft,2001); writeln(ft,201);
     for j:=-201 to 201 do writeln(ft,s[j]);
     close(ft);
end.
```

5) Probleme

1) Deduceți relația (3) pornind de la relațiile (1) și (2) și de la conservarea numărului de fotoni :

$$N = \sum_{i=1-g}^{m+g} E(i)\cdot\Delta f$$

Rezolvare

$$N = \sum_{i=1-g}^{m+g} E(i)\cdot\Delta f = \sum_{i=1-g}^{m+g} \sum_{j=-g}^{g} A(j)\cdot R(i-j)\cdot\Delta f\cdot\Delta f =$$

inversăm ordinea de sumare :

$$= \sum_{j=-g}^{g} A(j)\cdot\Delta f\cdot \sum_{i=1-g}^{m+g} R(i-j)\cdot\Delta f \qquad (8)$$

însă avem :

$$\sum_{i=1-g}^{m+g} R(i-j)\cdot\Delta f = \sum_{k=1-g-j}^{m+g-j} R(k)\cdot\Delta f = N$$

deoarece $g+j\ge 0$ și $g-j\ge 0$ înseamnă că indicele inferior al sumei va fi mai mic sau egal cu 1 , și indicele superior al sumei va fi mai mare sau egal cu m . In afara intervalului k=1..m avem R(k)=0 , iar suma pe k=1..m este N , comform (1).

Simplificând cu N ecuația (8) se obține ecuația (3) .

2) De obicei spectrul aparatului este o gausiană , iar spectrul real este o lorenziană , care deseori poate fi aproximată tot cu o gausiană , diferențe apărând doar în cozi .

Să se demonstreze că o gausiană (centrată pe zero , cu abaterea σ_1) convolutată cu o altă gausiană (cu abaterea σ_2) dă tot o gausiană (centrată pe zero) , cu abaterea : $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$

<u>Rezolvare</u>

$$f_1(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma_1}\cdot\exp(-\frac{x^2}{2\sigma_1^2})$$

$$f_2(z) = \frac{1}{\sqrt{2\pi}\cdot\sigma_2}\cdot\exp(-\frac{z^2}{2\sigma_2^2})$$

$$f(x) = \int_{-\infty}^{\infty} f_1(x-z)\cdot f_2(z)\cdot dz =$$

$$= \frac{1}{2\pi\sigma_1\sigma_2}\cdot\int_{-\infty}^{\infty}\exp(-\frac{(x-z)^2}{2\sigma_1^2} - \frac{z^2}{2\sigma_2^2})\cdot dz =$$

$$= \frac{1}{2\pi\sigma_1\sigma_2}\cdot\int_{-\infty}^{\infty}\exp(-\frac{z^2(\sigma_1^2+\sigma_2^2)-2zx\sigma_2^2+x^2\sigma_2^2}{2\sigma_1^2\sigma_2^2})\cdot dz =$$

realizăm un pătrat la exponențială :

$$= \frac{1}{2\pi\sigma_1\sigma_2}\cdot\int_{-\infty}^{\infty} dz\cdot$$

$$\cdot\exp(-\frac{\left[z\sqrt{\sigma_1^2+\sigma_2^2} - \frac{x\sigma_2^2}{\sqrt{\sigma_1^2+\sigma_2^2}}\right]^2 + x^2\sigma_2^2 - \frac{x^2\sigma_2^4}{\sigma_1^2+\sigma_2^2}}{2\sigma_1^2\sigma_2^2})$$

schimbare de variabilă :

$$y = z\sqrt{\sigma_1^2+\sigma_2^2} - \frac{x\sigma_2^2}{\sqrt{\sigma_1^2+\sigma_2^2}}$$

$$f(x) = \frac{1}{2\pi\sigma_1\sigma_2}\cdot\exp(-\frac{x^2}{2(\sigma_1{}^2+\sigma_2{}^2)})\cdot$$

$$\cdot\int_{-\infty}^{\infty}\exp(-\frac{y^2}{2\sigma_1{}^2\sigma_2{}^2})\cdot\frac{dy}{\sqrt{\sigma_1{}^2+\sigma_2{}^2}} =$$

$$= \frac{1}{\sqrt{2\pi}}\cdot\frac{1}{\sqrt{\sigma_1{}^2+\sigma_2{}^2}}\cdot\exp(-\frac{x^2}{2(\sigma_1{}^2+\sigma_2{}^2)})\cdot$$

$$\cdot\frac{1}{\sqrt{2\pi}\cdot\sigma_1\sigma_2}\cdot\int_{-\infty}^{\infty}\exp(-\frac{y^2}{2\sigma_1{}^2\sigma_2{}^2})\cdot dy =$$

Recunoaștem relația de normare a distribuției Gauss , pentru o distribuție cu abaterea $\sigma_1\sigma_2$:

$$\frac{1}{\sqrt{2\pi}\cdot\sigma_1\sigma_2}\cdot\int_{-\infty}^{\infty}\exp(-\frac{y^2}{2\sigma_1{}^2\sigma_2{}^2})\cdot dy = 1$$

Deci obținem rezultatul cerut :

$$f(x) = \frac{1}{\sqrt{2\pi}}\cdot\frac{1}{\sqrt{\sigma_1{}^2+\sigma_2{}^2}}\cdot\exp(-\frac{x^2}{2(\sigma_1{}^2+\sigma_2{}^2)})$$

BIBLIOGRAFIE

pentru algoritmi si metode numerice

1) C.I.Nistor : Prelucrarea automată a datelor fizice . Univ.Buc. , Fac.Fiz. , 1979
2) Paul L.DeVries : A First Course in Computational Physics , Miami University , 1993
3) Titus Beu : Analiză numerică în Turbo Pascal . Micro Informatica , Biblioteca PC , Cluj-Napoca , 1992
4) D.Demidovitch , I.Maron : Eléments de calcul numérique . Ed.Mir , 1979
5) M.Tiron : Prelucrarea statistică şi informaţională a datelor de măsură . Ed.Tehnică , 1976

pentru notiunile generale de matematica si fizica

6) Colectivul Catedrei de Fizică Atomică şi Nucleară : Lucrări practice de fizică nucleară . Univ.Buc. , Fac.Fiz. , 1987
7) Cursul de Fizică Berkeley : Fizică statistică (vol 5) . Ed.Didactică şi Pedagogică , 1983
8) M.Vraciu : Algebră liniară şi elemente de geometrie . Univ.Buc. , Fac.Fiz. , 1989

pentru limbajul Turbo Pascal

9) A.Constantinescu : Noţiuni de programare în Turbo Pascal 6.0 (note de curs) . Univ.Buc. , Fac.Fiz. , 1996
10) W.Kassera , V.Kassera : Programarea în Turbo Pascal 6.0 . Ed. Tipomur , 1992
11) Ghid de utilizare Turbo Pascal 5.0-5.5 . Micro Informatica , Cluj-Napoca , 1991
12) T.Bălănescu , şi alţii : Pascal şi Turbo Pascal , vol 1 şi 2 . Ed.Tehnică , 1992

Tiparul s-a efectuat sub c-da nr. 238/1996
la Tipografia Editurii Universității București

ISBN 973–575–079–1

Lei 3640